

# **HIGH-PERFORMANCE PARALLEL INTERFACE - 6400 Mbit/s Physical Layer (HIPPI-6400-PH)**

October 24, 1996

Secretariat:

Information Technology Industry Council (ITI)

**ABSTRACT:** This standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of user data at 6400 Mbit/s, per direction, over parallel copper cables across distances of 50 m, or over parallel fiber-optic cables across distances of 200 m. Small fixed-size micropackets provide an efficient, low-latency, structure for small transfers, and a component for large transfers.

**NOTE:**

*This is an internal working document of X3T11, a Technical Committee of Accredited Standards Committee X3. As such, this is not a completed standard. The contents are actively being modified by X3T11. This document is made available for review and comment only. For current information on the status of this document contact the individuals shown below:*

**POINTS OF CONTACT:**

Roger Cummings (X3T11 Chairman)  
Distributed Processing Technology  
140 Candace Drive  
Maitland, FL 32751  
(407) 830-5522 x348, Fax: (407) 260-5366  
E-mail: cummings\_roger@dpt.com

Carl Zeitler (X3T11 Vice-Chairman)  
IBM Corporation, MS 9440  
11400 Burnet Road  
Austin, TX 78758  
(512) 838-1797, Fax: (512) 838-3822  
E-mail: zeitler@ausvm6.vnet.ibm.com

Don Tolmie (HIPPI-6400-PH Technical Editor)  
Los Alamos National Laboratory  
CIC-5, MS-B255  
Los Alamos, NM 87545  
(505) 667-5502, Fax: (505) 665-7793  
E-mail: det@lanl.gov

## Comments on Rev 0.71

This document is mainly HIPPI-6400-PH 0.65 without the Scheduled Transfers. The change list, and single margin bars describing the changes between HIPPI-6400-PH Rev 0.6 and Rev 0.65 are retained. Changes between HIPPI-6400-PH Rev 0.65 and Rev 0.7 are marked with double margin bars. Changes to make Rev 0.71 are marked with a bold single margin bar.

This is a preliminary document undergoing lots of changes. Many of the additions are just place holders, or are put there to stimulate discussion. Hence, do not assume that the items herein are correct, or final – everything is subject to change. This page tries to outline where we are; what has been discussed and semi-approved, and what has been added or changed recently and deserves your special attention. This summary relates to changes since the previous revision. Also, previous open issues are outlined with a single box, new open issues ones are marked with a double bar on the left edge of the box.

Changes are marked with change bars in the margin. Minor changes, e.g., capitalization or spelling, did not warrant a change bar unless there were other substantive changes in the paragraph. The list below just describes the major changes, for detail changes please compare this revision to the previous revision.

Please help us in this development process by sending comments, corrections, and suggestions to the Technical Editor, Don Tolmie, of the Los Alamos National Laboratory, at [det@lanl.gov](mailto:det@lanl.gov). If you would like to address the whole group working on this document, send the Message to [hippi@network.com](mailto:hippi@network.com).

1. In the Abstract, Foreword, Introduction, and Scope, changed from "small Messages" to "small transfers", "building Block" to "component", and "large Messages" to "large transfers".
2. Removed the last line from the Abstract which read "Services are provided for transporting data associated with other HIPPI protocols.". This same change was made in the Foreword, Introduction, and Scope.
3. In the Introduction and Scope, removed the bullet items referring to Scheduled Transfers.

4. In the Introduction, changed the 5th bullet from "...and a building block for large Messages." to "...and a component for large transfers.".
5. Removed normative reference to X3.183, HIPPI-PH, since it was not referenced in the document.
6. Deleted definitions for Block, HIPPI-PH, Scheduled Transfer, Transfer, and Virtual Connection. Deleted HIPPI-PH reference in the "HIPPI port" definition.
7. In 4.1, changed "...16 (or fewer) bits wide for a fiber implementation." to "...eight bits wide for a fiber implementation.".
8. In 4.3, changed "...a micropacket is transmitted every 40 ns." to "...a micropacket is transmitted every 40 ns, with Null micropackets transmitted when other micropackets are not available.".
9. Deleted clause "4.5 Scheduled Transfers". Renumbered following subclauses to fill in the gap.
10. In 4.11, added the parenthetical phrase "(e.g., by the size of the TSEQ, RSEQ, and CR parameters)" to explain what about the protocol limited the maximum distance. Added the last sentence about a wide optical interface being described in an annex.
11. In 6.1, swapped the fields for TSEQ and RSEQ. Moved Tail to bit 06, Error to bit 07, TYPE to bits 02-05. Changed both the text and figure 7.
12. In 6.2, deleted the phrase "and may be used for Scheduled Transfers" in three places. Changed "plus two micropackets for Message headers (64 bytes)." to "plus a 32-byte Header micropacket." in 3 places.
13. In 6.3.2, changed "...shall be gap-fillers, used to keep the link active..." to "...are gap-fillers, and shall be used to keep the link active...".
14. In Table 2, changed the TSEQ entries for Reset/Initialize and Null from "1\*" to "x'FF'".
15. In 6.5, added last sentence reading "The Source Credit Counter range shall be 255, and the number of outstanding credits shall be  $\leq 255$ ."

16. In 6.5, changed Note 3 from "It is suggested that the VC with the least buffer space available be given priority to send Credit updates." to "If the Destination does not send adequate credits then the Source may not be able to send on every VC."
17. In 6.6.2, deleted "and figure 3" from the sentence "Refer to tables 3 and 4, and figure 3, for the transmission sequence."
18. In 6.6.2.1, replace two "it's" with what they were referring to.
19. In 6.6.3, deleted "and figure 3" from the sentence "Refer to tables 3 and 4, and figure 3, for the transmission sequence."
20. In the first paragraph of 7, changed "...will be padded..." to "...shall be padded..."
21. In 7, second paragraph, replaced the sentence referring to the Schedule Header with the sentence reading "The last eight bytes of the Header may be used by other protocols, and are not defined in this standard."
22. In Figure 10, removed the 2nd micropacket and labeled the last 8 bytes of the Header micropacket as "Opaque payload". Deleted the "(on all Messages)" entries. Changed the figure title from "Message headers" to "Header micropacket contents".
23. In the 2nd paragraph of 7, changed "...is part of a Scheduled Transfer..." to "...is part of a Scheduled Transfer (see 7.2)..."
24. In 7.2, changed "Scheduled Transfer" to "Scheduled Transfer as specified in xxx".
25. In the last item of 7.1, changed "...in bytes, of the data bytes following..." to "...in bytes, following..."
26. In 7.2, changed "...codings of the EtherType field shall be:" to "...codings of the EtherType field shall be as assigned in the current "Assigned Numbers" RFC. For the convenience of the reader, HIPPI-6400-specific EtherTypes are listed below:"
27. In 7.2, added "(See annex A.)" to the HIPPI-FP reference. Added EtherTypes x'8182' and x'8183' as "Locally administered" and "Reserved" respectively.
28. Deleted 7.3, the rest of clause 7, and all of original clause 8, i.e., everything defining the Scheduled Transfer. Renumbered the clauses, figures, tables, and annexes to fill in the gaps (these changes are not marked with margin bars in the document text).
29. Added new subclause 7.3 about opaque payloads.
30. In 8.2, added references to 8.4 for retransmission.
31. In 8.4, added the word "Micropacket" in the title. Added the first paragraph requiring 2 training sequences before a retransmission.
32. In 9.1.2, reworded the text and added a test for TSEQ  $\neq$  x'FF' and TYPE  $<$  x'8'.
33. In 9.1.4, added that "intermediate" Destinations shall treat unspecified TYPES as Data micropackets, and left it open for Final Destinations. Added the note giving an explanation for the Undefined TYPES treatment.
34. In 12.3, 4th bullet, changed the Link Reset or Initialize timeout from 10 seconds to 2 seconds. In the last paragraph changed from "...administrative action is required..." to "...administrative action may be required...". Added a new paragraph describing the Hold-off timer. Added the Hold-off timer to table 7.
35. In 15.2, changed the low level output from 0 V to 0.08 V.
36. In 15.3, changed the differential voltage from 50 mV to 150 mV.
37. In 15.4, changed "table 10" to "figure 17", and  $\leq 10$  A to  $\geq 5$  A.
38. Changed the "Signal path" figure from figure 19 to figure 16.
39. In 15.4, added "all appropriate compliance tests." to the EMI/RFI paragraph. Changed the test for the near end crosstalk.
40. In 15.5, added metric equivalents for the inch dimensions, and changed from "CL-2/FT4" to "CL-2.2/FT6".
41. Changed the title for the connector layout to "Bulkhead connector pin assignments". Changed it from a table to a figure. Changed the Power and Ground signals to "Reserved".
42. Changed the title of Annex A to "Mapping other HIPPI protocols over HIPPI-6400".
43. Deleted Annex A describing mapping to other HIPPI protocols, and renumbered Annex B as Annex A.

- 44. In A.2, changed "...5  $\mu$ s..." to "...20  $\mu$ s...".  
(A math error)
- 45. In A.4 - d, added that no registers are clocked in this step.
- 46. In figure A.4 moved the upper boxes apart so that the input lines coming from the bottom are shown more clearly.
- 47. Added A.5 giving the results of the CRC simulations.
- 48. The text for Annex C was removed since it was out of date and would probably create more confusion than clarity. Updated text will be included in future revisions.
- 49. Deleted Annex C.

# Contents

Page

Foreword.....	ix
Introduction.....	x
1 Scope.....	1
2 Normative references.....	1
3 Definitions and conventions.....	2
3.1 Definitions.....	2
3.2 Editorial conventions.....	2
3.3 Acronyms and other abbreviations.....	3
4 System overview.....	3
4.1 Links.....	3
4.2 Virtual channels.....	3
4.3 Micropacket.....	4
4.4 Message.....	5
4.5 FRAME and CLOCK signals.....	5
4.6 Flow control.....	5
4.7 Retransmission.....	6
4.8 Check functions.....	6
4.9 Copper physical layer (optional).....	6
4.10 Fiber physical layer (optional).....	6
5 Service interface.....	8
5.1 Service primitives.....	8
5.2 Sequences of primitives.....	8
6 Micropacket contents.....	9
6.1 Bit and byte assignments.....	9
6.2 Virtual channel (VC) selector.....	9
6.3 Micropacket TYPEs.....	10
6.4 Sequence number parameters.....	10
6.5 Credit update parameters.....	11
6.6 Check functions.....	12
7 Message structure.....	14
7.1 MAC Header.....	14
7.2 LLC/SNAP header.....	14
7.3 Opaque payload.....	14
8 Source specific operations.....	15
8.1 Credit update indications on Source side.....	15
8.2 ACK indications on Source side.....	15
8.3 ACKs and credit updates to far end.....	15
8.4 Micropacket retransmissions.....	15
9 Destination specific operations.....	16
9.1 Link level processing.....	16
9.2 Check for Message protocol errors.....	16
9.3 Generating ACKs.....	17

10	Signal line encoding.....	18
10.1	Signal line bit assignments.....	18
10.2	Source-side encoding for dc balance.....	20
10.3	Destination-side decoding.....	20
10.4	FRAME signal.....	21
11	Link training.....	21
12	Initialization and Shutdown.....	23
12.1	Link Reset.....	23
12.2	Initialize.....	23
12.3	Link Shutdown.....	23
13	Maintenance and control features.....	25
13.1	Logged errors.....	25
13.2	Timeouts.....	25
14	Timing.....	25
14.1	Source CLOCK signals.....	25
14.2	Destination CLOCK signals.....	25
14.3	FRAME, Data, and Control signals.....	27
15	Copper interface (optional).....	27
15.1	General.....	27
15.2	Electrical output interface.....	27
15.3	Electrical input interface.....	27
15.4	Electrical connector.....	27
15.5	Cable specifications.....	28
16	Optical interface (optional).....	28
16.1	General.....	28
16.2	Optical output interface.....	28
16.3	Optical input interface.....	28
16.4	Optical connector.....	28
16.5	Optical cable specifications.....	28
xx	Other Open Issues.....	29

## Tables

Table 1	– CRC coverages in a 128-byte Message.....	7
Table 2	– Micropacket contents summary.....	11
Table 3	– Signal line bit assignments in a 16-bit system.....	18
Table 4	– Signal line bit assignments in an 8-bit system.....	19
Table 5	– 4b/5b line coding.....	20
Table 6	– Summary of logged errors.....	26
Table 7	– Summary of timeouts.....	26
Table A.1	– Parallel LCRC input bits.....	31
Table A.2	– Parallel ECRC input bits.....	32
Table A.3	– 16-bit LCRC generator equations.....	33
Table A.4	– 64-bit LCRC generator equations.....	34
Table A.5	– 80-bit LCRC checker equations.....	35
Table A.6	– 64-bit ECRC generator / checker equations.....	36

## Figures

Figure 1 – System overview.....	4
Figure 2 – HIPPI-6400-PH link showing signal lines.....	4
Figure 3 – Logical micropacket format and naming conventions.....	5
Figure 4 – Message format.....	5
Figure 5 – Reverse direction control information.....	7
Figure 6 – HIPPI-6400-PH service interface.....	8
Figure 7 – Control bits summary.....	9
Figure 8 – LCRC implementation example.....	13
Figure 9 – ECRC implementation example.....	13
Figure 10 – Header micropacket contents.....	14
Figure 11 – 16-bit system micropacket.....	21
Figure 12 – 8-bit system micropacket.....	21
Figure 13 – 16-bit system training sequence.....	22
Figure 14 – 8-bit system training sequence.....	22
Figure 15 – Initialize and Link Reset operations.....	24
Figure 16 – Signal path.....	27
Figure 17 – Bulkhead connector pin assignments.....	29
Figure A.1 – Encode / decode circuit example.....	30
Figure A.2 – Parallel LCRC generator example.....	31
Figure A.3 – Parallel LCRC checker example.....	32
Figure A.4 – Parallel ECRC example.....	33

## Annex

A Implementation comments.....	30
A.1 4b/5b encoding and decoding.....	30
A.2 Frequency differences between Source and Destination.....	30
A.3 LCRC parallel implementation.....	31
A.4 ECRC parallel implementation.....	32
A.5 Undetected errors.....	32





**Foreword** (This foreword is not part of American National Standard X3.xxx-199x.)

This American National Standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of user data at 6400 Mbit/s, per direction, over parallel copper cables across distances of 50 m, or over parallel fiber-optic cables across distances of 200 m. Small fixed-size micropackets provide an efficient, low-latency, structure for small transfers, and a component for large transfers.

This standard provides an upward growth path for legacy HIPPI-based systems.

This document includes one annex which is informative and is not considered part of the standard.

Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to the X3 Secretariat, Information Technology Industry Council, 1250 Eye Street, NW, Suite 200, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by Accredited Standards Committee on Information Processing Systems, X3. Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, the X3 Committee had the following members:

(List of X3 Committee members to be included in the published standard by the ANSI Editor.)

Subcommittee X3T11 on Device Level Interfaces, which developed this standard, had the following participants:

(List of X3T11 Committee members, and other active participants, at the time the document is forwarded for public review, will be included by the Technical Editor.)

## Introduction

This American National Standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of user data at 6400 Mbit/s, per direction, over parallel copper cables across distances of 50 m, or over parallel fiber-optic cables across distances of 200 m. Small fixed-size micropackets provide an efficient, low-latency, structure for small transfers, and a component for large transfers.

Characteristics of a HIPPI-6400-PH physical-layer interface include:

- User data transfer bandwidth of 6400 Mbit/s (800 MByte/s).
- A full-duplex link capable of independent full-bandwidth transfers in both directions simultaneously.
- Four virtual circuits providing a limited multiplexing capability.
- A fixed size transfer unit, i.e., a 32-byte micropacket, for hardware efficiency.
- A small transfer unit resulting in low latency for short Messages, and a component for large transfers.
- Credit-based flow control that prevents buffer overflow.
- End-to-end, as well as link-to-link, check sums.
- Automatic retransmission of errored data providing guaranteed, in-order, reliable, data delivery.
- An ac coupled parallel electrical interface for limited distance applications, and a parallel fiber-optic interface for longer distances.
- Support for carrying legacy HIPPI-800 and HIPPI-1600 traffic.
- A physical-layer interface continuing the HIPPI tradition of simplicity.

## American National Standard for Information Technology –

# High-Performance Parallel Interface – 6400 Mbit/s Physical Layer (HIPPI-6400-PH)

## 1 Scope

This American National Standard specifies a physical-level, point-to-point, full-duplex, link interface for reliable, flow-controlled, transmission of user data at 6400 Mbit/s, per direction, over parallel copper cables across distances of 50 m, or over parallel fiber-optic cables across distances of 200 m. Small fixed-size micropackets provide an efficient, low-latency, structure for small transfers, and a component for large transfers.

Specifications are included for:

- automatic retransmission of errored data;
- the format of a small data transfer unit called a micropacket;
- a Message structure that includes routing information for network applications;
- end-to-end, as well as link-to-link, check sums;
- the timing requirements of the parallel signals;
- a parallel interface using copper coaxial cable;
- a parallel interface using ribbon fiber-optic cable;
- a link-level protocol tuned for a maximum distance of 1 km.

## 2 Normative references

The following American National Standards contain provisions which, through reference in this text, constitute provisions of this American National Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

ANSI X3.210-1992, *High-Performance Parallel Interface, Framing Protocol (HIPPI-FP)*

ANSI X3.xxx-199x, *High-Performance Parallel Interface, 6400 Mbit/s Switch Control (HIPPI-6400-SC)*

ANSI/IEEE Std 802-1990, *IEEE Standards for Local and Metropolitan Area Networks: Overview and architecture (formerly known as IEEE Std 802.1A, Project 802: Local and Metropolitan Area Network Standard — Overview and Architecture)*

ISO/IEC 8802-2:1989 (ANSI/IEEE Std 802.2-1989), *Information Processing Systems – Local Area Networks – Part 2: Logical link control*

### 3 Definitions and conventions

#### 3.1 Definitions

For the purposes of this standard, the following definitions apply.

**3.1.1 acknowledge (ACK):** Confirmation that the Destination has received the micropacket without errors.

**3.1.2 administrator:** A station management entity providing external management control.

**3.1.3 credit:** A credit corresponds to one micropacket's worth of buffer space available in the Destination's VC buffer.

**3.1.4 Destination:** The equipment that receives the data.

**3.1.5 Final Destination:** The equipment that receives, and operates on, the payload portion of the micropackets. This is typically a host computer system, but may also be a translator, bridge, or router.

**3.1.6 HIPPI port:** A HIPPI-6400-PH Source or Destination.

**3.1.7 link:** A full-duplex connection between HIPPI-6400-PH devices.

**3.1.8 log:** The process of incrementing a specific event counter upon detection of the event, e.g., logging an ECRC error.

**3.1.9 Message:** An ordered sequence of one or more micropackets that have the same VC. The first micropacket is a Header micropacket. The last micropacket, which may also be the first micropacket, has the TAIL bit set. (See 4.4.)

**3.1.10 micropacket:** The basic transfer unit consisting of 32 data bytes and 64 bits of control information.

**3.1.11 optional:** Characteristics that are not required by HIPPI-6400-PH. However, if any optional characteristic is implemented, it shall be implemented as defined in HIPPI-6400-PH.

**3.1.12 Originating Source:** The equipment that generates the payload portion of the micropackets. This is typically a host computer system, but may also be a translator, bridge, or router.

**3.1.13 Source:** The equipment that transmits the data.

**3.1.14 syndrome:** The value (should be zero) obtained by exclusive ORing the calculated CRC value with the CRC value received with the micropacket.

**3.1.15 Universal LAN MAC Address (ULA):** The 48-bit MAC address specified by the IEEE 802 Overview Standard.

**3.1.16 Virtual Channel (VC):** One of four logical paths within each direction of a link.

#### 3.2 Editorial conventions

In this standard, certain terms that are proper names of signals or similar terms are printed in uppercase to avoid possible confusion with other uses of the same words (e.g., FRAME). Any lowercase uses of these words have the normal technical English meaning.

A number of conditions, sequence parameters, events, states, or similar terms are printed with the first letter of each word in uppercase and the rest lowercase (e.g., Block, Source). Any lowercase uses of these words have the normal technical English meaning.

The word *shall* when used in this American National standard, states a mandatory rule or requirement. The word *should* when used in this standard, states a recommendation.

### 3.2.1 Binary notation

Binary notation is used to represent relatively short fields. For example a two-bit field containing the binary value of 10 is shown in binary format as b'10'.

### 3.2.2 Hexadecimal notation

Hexadecimal notation is used to represent some fields. For example a two-byte field containing a binary value of b'11000100 00000011' is shown in hexadecimal format as x'C403'.

### 3.3 Acronyms and other abbreviations

<b>ACK</b>	acknowledge indication
<b>CR</b>	credit amount parameter
<b>CRC</b>	cyclic redundancy check
<b>ECRC</b>	end-to-end CRC
<b>HIPPI</b>	High-Performance Parallel Interface
<b>LCRC</b>	link CRC
<b>lsb</b>	least significant bit
<b>MAC</b>	Media Access Control
<b>msb</b>	most significant bit
<b>ns</b>	nanoseconds
<b>ps</b>	picoseconds
<b>RSEQ</b>	receive sequence number
<b>TSEQ</b>	transmit sequence number
<b>ULA</b>	Universal LAN MAC Address
<b>ULP</b>	upper-layer protocol
<b>VC</b>	virtual channel
<b>VCR</b>	virtual channel Credit selector
<b>μs</b>	microseconds
<b>Ω</b>	ohms

## 4 System overview

This clause provides an overview of the structure, concepts, and mechanisms used in HIPPI-6400-PH. Figure 1 gives an example of a HIPPI-6400 system.

### 4.1 Links

HIPPI-6400-PH defines a point-to-point physical link for transferring micropackets. The physical links, as shown in figure 2, are bi-directional. The logical links are simplex, i.e., the data inbound and outbound are completely separate. Some control information, e.g., credit, flows in the reverse direction, and it is included in the micropackets flowing in the reverse direction. This is why the physical links must be bi-directional with information flowing in both directions simultaneously.

A link is composed of two Sources that transmit information, and two Destinations that receive information. Each end of a link has a Source and a Destination.

The data path is 16 bits wide for a copper implementation, and is eight bits wide for a fiber implementation. The control path is one-fourth the width of the data path, e.g., the control path for a copper implementation would be 4 bits wide. CLOCK, CLOCK\_2, and FRAME are individual signals carried on separate conductors. The CLOCK\_2 signal is only used in 16-bit systems.

### 4.2 Virtual channels

Four virtual channels, VC0, VC1, VC2, and VC3 are available in each direction on each link. The VCs are assigned to specific Message sizes and transfer methods. All of the micropackets of a Message are transmitted on a single VC, i.e., the VC number does not change as the micropackets travel from the Originating Source to the Final Destination over one or more links. The VCs provide a multiplexing mechanism which can be used to prevent a large Message from Blocking a small Message until the large Message has completed.

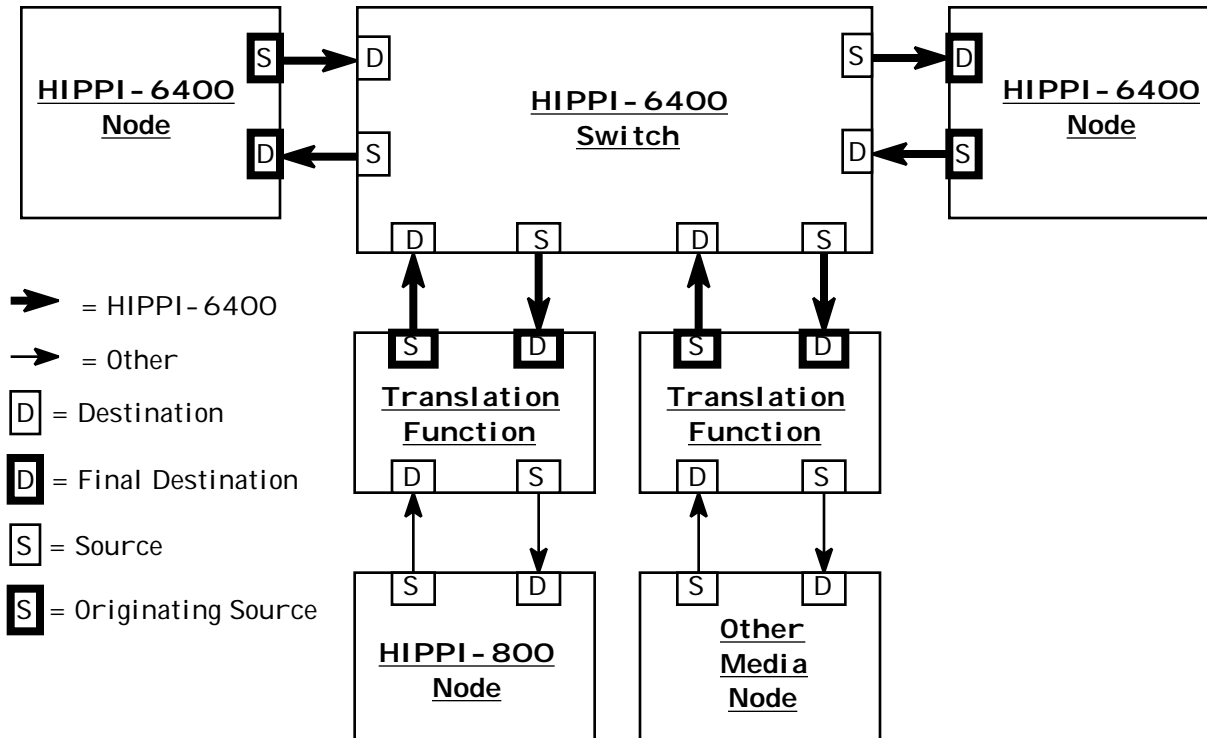
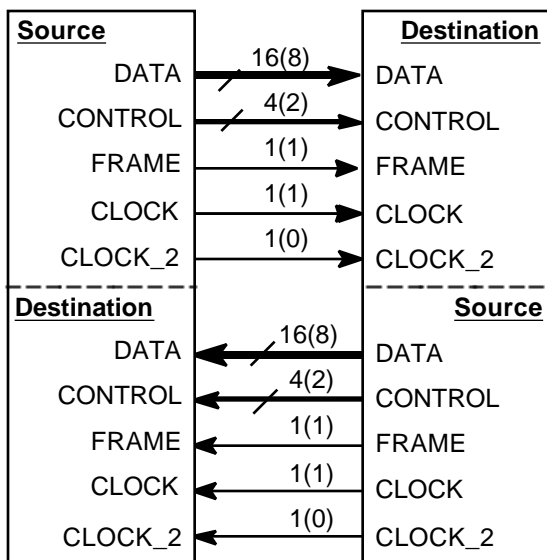


Figure 1 – System overview

#### 4.3 Micropacket



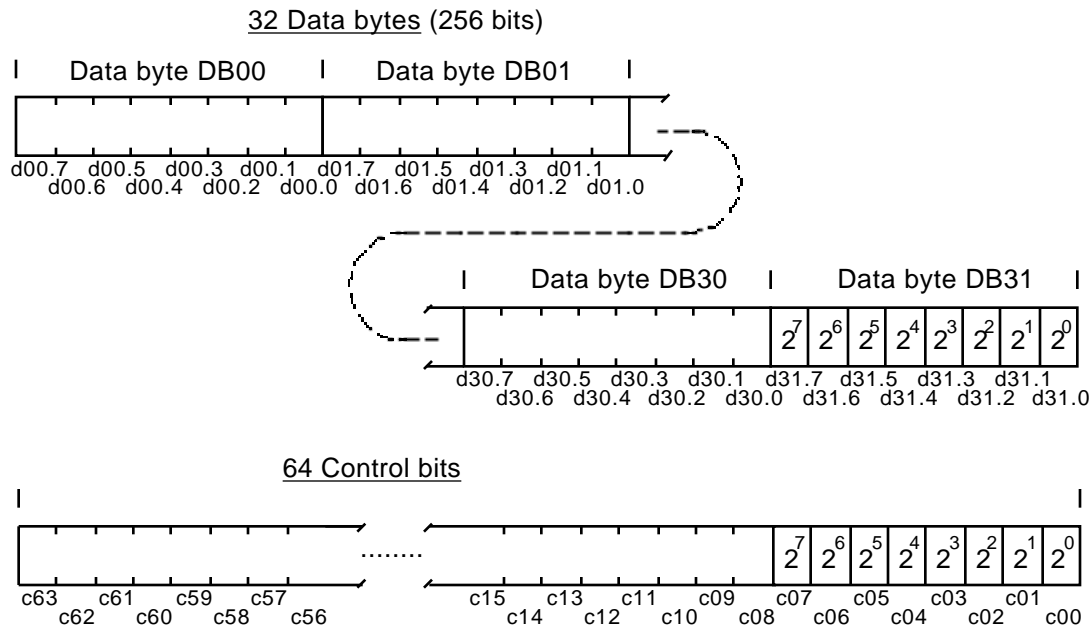
(Numbers in parenthesis are for an 8-bit system.  
CLOCK\_2 is only used in 16-bit systems.)

Figure 2 – HIPPI-6400-PH link showing signal lines

Micropackets are the basic transfer unit. As shown in figure 3 a micropacket is composed of 32 data bytes and 64 bits of control information. At 6400 Mbit/s a micropacket is transmitted every 40 ns, with Null micropackets transmitted when other micropackets are not available. Credit and retransmit operations are performed on a micropacket basis.

The 64 bits of control information in each micropacket includes parameters for:

- selecting a VC;
- detecting missing micropackets;
- denoting the types of information in the micropacket;
- marking the last micropacket of a Message;
- signalling that the Message was truncated at its originator, or damaged en-route;
- passing credit information from the Destination to the Source;
- Link-level and end-to-end checksums.



Naming conventions:

Data bytes are labeled capital DB and a two-digit number, e.g., DB00.

In a parameter that uses multiple bytes, the most-significant byte is the lowest-numbered byte.

Data bits are labeled lower case d, a two-digit byte number, and a one-digit bit number, e.g., d31.7.

Control bits are labeled lower case c and a two-digit number, e.g., c00.

In a parameter that uses multiple bits, the most-significant bit is the highest-numbered bit.

**Figure 3 – Logical micropacket format and naming conventions**

**4.4 Message**

As shown in figure 4, a Message is an ordered sequence of one or more micropackets which have the same VC. The first micropacket of a Message, i.e., the Header micropacket, contains information used to route through a HIPPI-6400 fabric. The last micropacket of the Message is marked with the TAIL bit.

Micropacket Transmission order	1	Header information	c63–c00
	2	1st 32 bytes of Message data	c63–c00
	3	2nd 32 bytes of Message data	c63–c00
	...	...	...
	n	Last bytes of Message data	c63–c00

**Figure 4 – Message format**

**4.5 FRAME and CLOCK signals**

The FRAME signal, carried on a separate signal line, marks a micropacket's beginning. Both edges of either the CLOCK or CLOCK\_2 signals, also carried on separate signal lines, are used for strobing the data. The data, control, and FRAME signals from a Source are synchronous with that Source's CLOCK and CLOCK\_2 signals. The CLOCK rate is dependent on the width of the data bus, e.g., a 16-bit data bus utilizing 4b/5b encoding requires the CLOCK line to run at 250 MHz and each data and control line may transition every 2 ns.

**4.6 Flow control**

Credit-based flow control is used. As shown in figure 5, the credits are assigned on a VC basis, i.e., VC0's credits are separate from VC1's credits. The Destination end of a link

grants credits to match the number of free receive buffers for a particular VC. The Source consumes credits as it moves micropackets from the VC Buffers to the Output Buffer. Note that flow control is on a link basis, i.e., hop-by-hop.

#### 4.7 Retransmission

Go-back-N retransmission is used. The CRCs in each micropacket are checked at the Destination side of a link; at the Input Buffer in figure 5. Correct micropackets are acknowledged, incorrect micropackets are discarded. Retransmission of incorrect micropackets is automatic. Note that retransmission is independent of the VC used, and also independent of the credit information, i.e., retransmission occurs between the Output and Input Buffers in figure 5 while VC and credit information pertains only to the VC Buffers. Retransmission is on a link basis, i.e., hop-by-hop.

#### 4.8 Check functions

As shown in table 1, two 16-bit cyclic redundancy checks (CRCs) are used, and they use different polynomials. The end-to-end CRC (ECRC) covers the data bytes of all of the micropackets in a Message, i.e., the Header micropacket and all of the Data micropackets (if any) up to this point in a Message. The ECRC does not cover the control bits. The ECRC is unchanged from the Originating Source to the Final Destination. The ECRC is accumulated over an entire Message, i.e., it is not re-initialized for intermediate Data micropackets. (See 6.6.3.)

The link CRC (LCRC) covers all of the data and control bits of a micropacket, with the exception of itself. The LCRC is initialized for each micropacket, and must be calculated fresh for each link since other control fields change.

The combination of two 16-bit CRCs provides a stronger check than a single 16-bit CRC for link-level checking of individual micropackets. In addition, the 16-bit ECRC provides checking over a whole Message.

#### 4.9 Copper physical layer (optional)

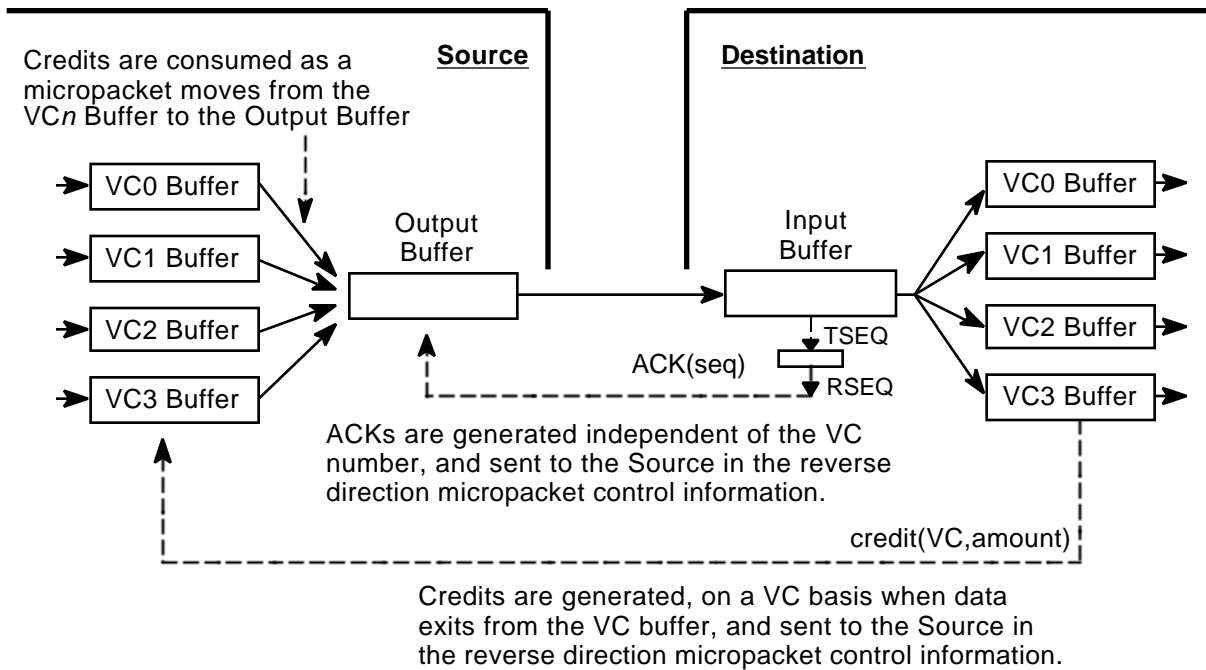
The optional HIPPI-6400-PH copper variant uses a cable with 46 conductor pairs, 23 in each direction, and an overall shield. The length is dependent upon the quality of the cable. The signals are ac coupled to the cable to accommodate some difference in the ground potential between the equipment. (See clause 15.)

#### 4.10 Fiber physical layer (optional)

The optional HIPPI-6400-PH fiber variant uses a fiber-ribbon cable with 12 multi-mode fibers in each direction. The length is limited to 1 km by the protocol, (e.g., by the size of the TSEQ, RSEQ, and CR parameters), and may be limited to shorter distances by the physical constraints of the optical system. (See clause 16.) A non-standard double-wide optical implementation is described in annex B.

<i>Open Issue – Someone from E-Systems needs to provide the text describing the double-wide optical interface.</i>
--



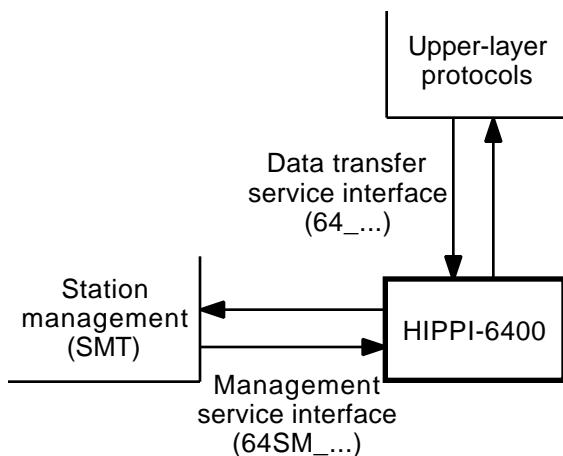
**Figure 5 – Reverse direction control information****Table 1 – CRC coverages in a 128-byte Message**

Micropacket number	Data Bytes DB00 – DB31 contents	ECRC coverage	LCRC coverage
1	Header	Header	Header, c00 – c47
2	Bytes 1 – 32	Header and Bytes 1 – 32	Bytes 1 – 32, c00 – c47
3	Bytes 33 – 64	Header and Bytes 1 – 64	Bytes 33 – 64, c00 – c47
4	Bytes 65 – 96	Header and Bytes 1 – 96	Bytes 65 – 96, c00 – c47
5	Bytes 97 – 128	Header and Bytes 1 – 128	Bytes 97 – 128, c00 – c47

## 5 Service interface

This clause specifies the services provided by HIPPI-6400-PH. The intent is to allow ULPs to operate correctly with this HIPPI-6400-PH. How many of the services described herein are chosen for a given implementation is up to that implementor; however, a set of HIPPI-6400-PH services must be supplied sufficient to satisfy the ULP(s) being used. The services as defined herein do not imply any particular implementation, or any interface.

Figure 6 shows the relationship of the HIPPI-6400-PH interfaces.



**Figure 6 – HIPPI-6400-PH service interface**

### 5.1 Service primitives

The primitives, in the context of the state transitions in clause 5, are declared required or optional. Additionally, parameters are either required, conditional, or optional. All of the primitives and parameters are considered as required except where explicitly stated otherwise.

HIPPI-6400-PH service primitives are of four types.

- *Request primitives* are issued by a service user to initiate a service provided by the HIPPI-6400-PH. In this standard, a second Request primitive of the same name shall not be issued until the Confirm for the first request is received.

- *Confirm primitives* are issued by the HIPPI-6400-PH to acknowledge a Request.

- *Indicate primitives* are issued by the HIPPI-6400-PH to notify the service user of a local event. This primitive is similar in nature to an unsolicited interrupt. Note that the local event may have been caused by a service Request. In this standard, a second Indicate primitive of the same name shall not be issued until the Response for the first Indicate is received.

- *Response primitives* are issued by a service user to acknowledge an Indicate.

### 5.2 Sequences of primitives

The order of execution of service primitives is not arbitrary. Logical and time sequence relationships exist for all described service primitives. Time sequence diagrams are used to illustrate a valid sequence. Other valid sequences may exist. The sequence of events between peer users across the user/provider interface is illustrated. In the time sequence diagrams the HIPPI-6400-PH users are depicted on either side of the vertical bars while the HIPPI-6400-PH acts as the service provider.

*NOTE - The intent is to flesh out the service primitives similar to what is in HIPPI-PH today.*

## 6 Micropacket contents

### 6.1 Bit and byte assignments

As shown in figure 3, each micropacket shall consist of 32 data bytes and 64 bits of control information. The data bytes shall be numbered DB00 - DB31. DB00 shall be transmitted first. The data bits in the micropacket shall be numbered dxx.y where xx is the byte number and y is the bit number in the byte.

The 64 bits of control information shall be numbered as bits c63 – c00. Control bit c00 shall be transmitted first. As shown in figure 3, a field with a numerical value shall have its most-significant bit in the highest numbered bit position.

The control information shall contain the following parameters located in the bits specified. The Source side of a link supplies all of the parameters, except for RSEQ, VCR and CR which come to the Source from its local Destination side. The VC parameter comes from the Originating Source. The TAIL, TYPE, and ECRC parameters normally come from the Originating Source, but may under error conditions come from an intermediate device (see 9.2.3 and 9.2.4).

VC (2 bits, c00–c01) – The virtual channel selector. (See 6.2.)

TYPE (4 bits, c02–c05) – Identifies the type of information within the micropacket. (See 6.3.)

TAIL (1 bit, c06) – TAIL = 1 identifies the last micropacket of a Message. TAIL = 0 means that more micropackets for this Message follow.

ERROR (1 bit, c07) – ERROR = 1 means that an unrecoverable error has been detected in the Message, do not check the ECRC. ERROR = 0 means that the Message is OK so far. (See 9.1.3.)

VCR (2 bits, c08–c09) – Virtual channel number associated with credit addition. (See 6.5.)

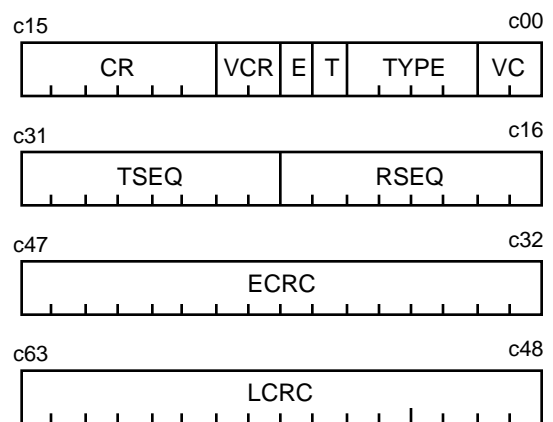
CR (6 bits, c10–c15) – Amount of credit to add to the virtual channel specified in VCR. (See 6.5.)

RSEQ (8 bits, c16–c23) – Sequence number associated with micropacket ACK indication. (See 6.4.)

TSEQ (8 bits, c24–c31) – Sequence number of transmitted micropacket. (See 6.4.)

ECRC (16 bits, c32–c47) – End-to-end checksum covering all of the data bytes up to this point in a Message, including those in the Header micropacket. (See 6.6.)

LCRC (16 bits, c48–c63) – Link level checksum covering the 32 data bytes, and the c00 through c47 control bits, in this micropacket. (See 6.6.)



Note – Transmission order is top to bottom, and right to left, in 4-bit groups, as shown in tables 3 and 4. The most-significant-bit of a parameter is at the left end of its field.

**Figure 7 – Control bits summary**

### 6.2 Virtual channel (VC) selector

Four virtual channels shall be available in each direction on a link. Messages on the virtual channels shall be assigned as follows:

- VC0 = Messages with a maximum size of 64 data micropackets (2048 bytes) plus a 32-byte Header micropacket.

- VC1 = Messages with a maximum size of 4096 data micropackets (128 KBytes) plus a 32-byte Header micropacket. Also carries Admin Request Messages.

- VC2 = Messages with a maximum size of 4096 data micropackets (128 KBytes) plus a 32-byte Header micropacket. Also carries Admin Response Messages.

- VC3 = Messages with a maximum size of 4 GBytes. Each Message shall contain a Header micropacket as the first micropacket of the Message.

### 6.3 Micropacket TYPEs

The 4-bit TYPE parameter shall indicate the contents of the micropacket. Unspecified TYPE values are reserved for future use. Actions to be taken as a result of receiving an unspecified TYPE are detailed in 9.1.4. Only micropackets whose TYPE  $\geq$  x'8' shall be retransmitted.

#### 6.3.1 TYPE = link control micropackets

Control micropackets operate at the link level, do not carry any user data, acknowledgments, or credit update information. (See clause 12.) Control micropackets include:

- Reset (TYPE = x'2') – Sent to initiate a Link Reset operation. (See 12.1.)
- Reset\_ACK (TYPE = x'3') – The receiving device has completed the Link Reset operation.
- Initialize (TYPE = x'4') – Sent to initiate an Initialization operation. (See 12.2.)
- Initialize\_ACK (TYPE = x'5') – The receiving device has completed the Initialization operation.

#### 6.3.2 TYPE = Null micropackets

Null micropackets (TYPE = x'7') are gap-fillers, and shall be used to keep the link active when there are no other micropackets to transmit. Null micropackets may carry ACK indications.

#### 6.3.3 TYPE = Data micropackets

Data micropackets (TYPE = x'8') carry user data, and may carry the Schedule Header.

#### 6.3.4 TYPE = Header micropackets

Header micropackets (TYPE = x'9') carry routing and control information.

#### 6.3.5 TYPE = Credit-only micropackets

When credits are available, and there are no Data micropackets to send, then Credit-only micropackets (TYPE = x'A') are used to carry credit update information, and acknowledgments.

#### 6.3.6 TYPE = Admin micropackets

Admin micropackets (TYPE = x'F') carry operations and maintenance information between the end devices of a link. Admin micropackets are fully specified in HIPPI-6400-SC.

*Open Issue – Admin micropackets will be defined in this document, and their usage described in HIPPI-6400-SC. This note is left as a reminder and place holder.*

### 6.4 Sequence number parameters

The transmit sequence number (TSEQ) shall increment by one for each micropacket transmitted whose TYPE  $\geq$  x'8'. TSEQ shall wrap from x'FE' to x'00'. The receive sequence number (RSEQ) shall be used to acknowledge (ACK) these micropackets. RSEQ shall equal the TSEQ of the most recent micropacket being acknowledged. TSEQ shall begin with the value = x'00' after a Link Reset. RSEQ = x'FF' indicates that no ACK indication is being transmitted (used while the link fills with micropackets after a Link Reset).

#### NOTES

- 1 The TSEQ and RSEQ parameters are independent of the virtual channel used to transmit the micropacket.

**Table 2 – Micropacket contents summary**

	Reset/ Initialize	Null	Credit-only	Header	Data	Admin
<b>Data Bytes contents</b>	0*	0*	0*	32 bytes of header information (see 7.1)	32 bytes of user data, or Schedule Parameters	Administrative information
<b>VC</b>	0*	0*	0*	any	any	Requests on VC1 Responses on VC2
<b>TYPE</b> (hex)	2,3,4,5	7	A	9	8	F
<b>TAIL</b>	1*	0*	0*	= 1 on last micropacket of Message	= 1 on last micropacket of Message	1
<b>ERROR</b>	0*	0*	0	= 1 if error	= 1 if error	= 1 if error
<b>TSEQ</b>	x'FF'	x'FF'	increments	increments	increments	increments
<b>RSEQ</b>	1*	ACK	ACK	ACK	ACK	ACK
<b>VCR</b>	0*	0*	any	any	any	any
<b>CR</b>	0*	0*	any	any	any	any
<b>LCRC</b>	single	single	single	single	single	single
<b>ECRC</b>	single	single	single	accumulating	accumulating	single
0* = transmit all bits of this field as 0's, a receiver must permit any value 1* = transmit all bits of this field as 1's, a receiver must permit any value any = any data value as appropriate single = this CRC is calculated and checked for this single micropacket accumulating = ECRC as defined in 6.6.3						

2 The TSEQ and RSEQ parameters are local to a specific link. For example, a micropacket that transverse more than one link will most likely have different TSEQ numbers on the different links.

3 The first micropacket with TYPE  $\geq$  x'8' following a stomped micropacket (see 6.6.2.1) uses the same TSEQ value as in the stomped micropacket since that TSEQ value was not consumed.

The wrap at x'FE' shall be taken into account when processing ACK indications. For example, if the previous ACK indication had RSEQ = x'F7', and an ACK indication with RSEQ = x'03' is received, then the micropackets whose TSEQ value = x'F8' through x'FE', and from x'00' through x'03', are acknowledged and their memory may be reused by the Source.

## 6.5 Credit update parameters

The Destination shall insert the VCR and CR parameters in micropackets to inform the Source that CR number of micropacket buffers have been freed up for the VC indicated by VCR. The Source shall increase its Credit Counter for this virtual channel by the value in CR. The Source Credit Counter range shall be 255, and the number of outstanding credits shall be  $\leq$  255.

### NOTES

1 The CR value is an incremental update value, not the number of buffers currently available in the Destination.

2 At 40 ns per micropacket, and 5 ns per meter of cable, each credit is equivalent to about 8 meters of cable. Hence, a Credit Count, and Destination buffer capacity per virtual channel, of 255 will

support full bandwidth on a 1 km link when round trip time is taken into account and Destination latency is low.

3 If the Destination does not send adequate credits then the Source may not be able to send on every VC.

## 6.6 Check functions

### 6.6.1 Intended use of CRCs

Two 16-bit cyclic redundancy checks (CRCs) shall be used. The link CRC (LCRC) checks all of the data bytes and control bits in a single micropacket. The LCRC shall be generated by a link's Source, and checked by the same link's Destination, i.e., it is local to a link.

The end-to-end CRC (ECRC) checks all of the data bytes of a Message, i.e., it may cover multiple micropackets. The ECRC shall be generated by the Originating Source, and should be passed unchanged through intermediate link-level devices. The ECRC shall be checked at each Destination in the path. See 9.1 for ECRC error operations.

While this standard covers the link level and host interface, other documents may require intermediate link-level devices to carry the ECRC across them, for example, across switches. Link-level devices, as described here, are devices that do not operate on the payload portion of data micropackets.

### 6.6.2 Link-level CRC (LCRC)

The link CRC (LCRC) shall cover all of the data bytes, and the control bits except for itself. The LCRC generator and checker shall be initialized to all ones (x'FFFF') for each micropacket.

The LCRC polynomial shall be:

$$x^{16} + x^{12} + x^5 + 1$$

Figure 8 is an example serial implementation. The LCRC may be implemented in a parallel fashion rather than serial, but must produce the same results as the serial example. The c63 through c48 bits are the LCRC bits in the control word. The incoming data and control

bits are exclusive OR'd with c48 to generate a *sum* value; the *sum* value is exclusive OR'd with selected control bits as they are shifted right once each bit period. The data and control bits shall be input to the generator in transmission sequence, i.e., 64 data bits, 16 control bits, 64 data bits, 16 control bits, etc. The sequence is d00.0, d00.1, d00.2, ...d00.7, d01.0...d01.7, ...d07.7, c00, c01, c02...c15, d08.0...d15.7, c16...c31, d16.0...d23.7, c32...c47, d24.0...d31.7. Refer to tables 3 and 4 for the transmission sequence. After passing all 304 input bits, c63-through-c48 contain the most-significant through least-significant bits of the LCRC.

At the destination, the LCRC check may be implemented by clocking the entire micropacket, including the LCRC parameter (c63..c48), into either a serial or parallel checker. In this case, a residue is available in the checker register after the last clock rather than a syndrome. If this check method is used, a residue of x'0000' indicates no errors, and x'06A9' indicates that a "stomp" code was received.

See 9.1.1 for details of a Destination's actions when checking the LCRC. See annex A.3 for the equations to generate the LCRC in a parallel fashion.

#### 6.6.2.1 Stomp code at Source

A Source may decide during the course of transmitting a micropacket that it wishes to "nullify" that transmission. This shall be done by XORing a "stomp" code of x'874D' with the LCRC that it has calculated for the micropacket. The Source shall treat a "stomped" micropacket as if it never occurred, i.e., not save the "stomped" micropacket in the retransmit buffer, and not increment the TSEQ number since the TSEQ number was not consumed.

NOTE – The stomp code of x'874D' was chosen as it has an equal number of 0's and 1's, resulting in flipping exactly half the bits when XOR'd with the LCRC.

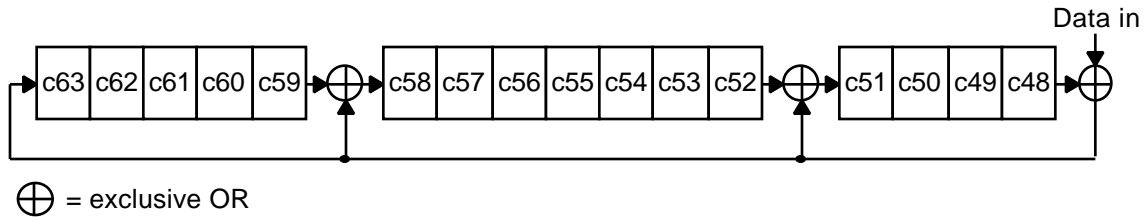


Figure 8 – LCRC implementation example

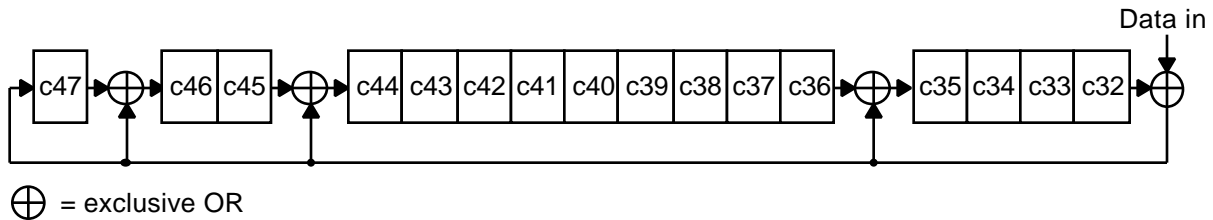


Figure 9 – ECRC implementation example

### 6.6.2.2 Stomp code at Destination

If the Destination detects a "stomp" code (see 6.6.2), then an LCRC error shall not be logged (see 9.1.1).

### 6.6.3 End-to-end CRC (ECRC)

The end-to-end CRC (ECRC) shall include only the micropacket's data bytes, not the control bits, in its calculation. The ECRC shall include all of a Message's data bytes up to this point in the Message, i.e., the data bytes in the Header micropacket and in all of the Data micropackets up to this point in the Message.

All Sources not generating the original ECRC shall check the ECRC prior to transmission, and if the ECRC is in error then set ERROR = 1 in this micropacket's control bits. An ECRC\_Source\_Error shall be logged for only the first occurrence of this error in a Message (see 13.1). This aids in error isolation and prevents endless retransmission loops.

The ECRC generator polynomial shall be:

$$x^{16} + x^{12} + x^3 + x + 1$$

The ECRC is calculated and maintained independently for each VC. The ECRC

checker and generator for a VC shall be initialized to all ones (x'FFFF') for each Message. Figure 9 is an example ECRC serial implementation. The ECRC may be implemented in a parallel fashion rather than serial, but must produce the same results as the serial example. The c47 through c32 bits are the ECRC bits in the control word. The incoming data bits are exclusive OR'd with c32 to generate a *sum* value; the *sum* value is exclusive OR'd with selected control bits as they are shifted right once each bit period. The data bits shall be input to the generator in transmission sequence, i.e., d00.0, d00.1, d00.2, ...d00.7, d01.0...d01.7, ...d31.7. Refer to tables 3 and 4 for the transmission sequence. After passing all 256 of the micropacket's data bits, c47-through-c32 contain the most-significant through least-significant bits of the ECRC for this micropacket. The ECRC value will normally be different for each micropacket of a Message since the ECRC accumulates as the Message progresses (see table 1).

See 9.1.3 for details of a Destination's actions when checking the ECRC. See annex A.4 for the equations to generate the ECRC in a 64-bit-wide fashion.

## 7 Message structure

As defined in 4.4, a Message is an ordered sequence of one or more micropackets which have the same VC, start with a Header micropacket (TYPE = Header), and have TAIL = 1 in the last micropacket. Each VC may only have a single Message in progress at any time. Since only complete micropackets are transmitted, a Message that is not an integral number of 32 bytes in length shall be padded in the last micropacket.

The Message header format is shown in figure 10 as a group of 32-bit words. The Media Access Control (MAC) Header, and LLC/SNAP header, shall reside in the first 24 bytes of all Header micropackets. The last eight bytes of the Header may be used by other protocols, and are not defined in this standard. If a parameter uses more than one byte, the lowest numbered byte is the most-significant byte.

### 7.1 MAC Header

The MAC header shall be included in all HIPPI-6400 Messages. The MAC header shall be in the first micropacket (Header micropacket) of a Message, and shall contain:

D\_ULA (48 bits, DB00-DB05) – The IEEE 48-bit ULA network address, as defined in ANSI/IEEE Std 802, identifying the payload's Destination.

S\_ULA (48 bits, DB06-DB11) – The IEEE 48-bit ULA network address, as defined in ANSI/IEEE Std 802, identifying the payload's Source.

M\_len (32 bits, DB12-DB15) – The Message length, in bytes, following the M\_len field, exclusive of any padding in the last micropacket.

### 7.2 LLC/SNAP header

The LLC/SNAP header, as defined in ISO/IEC 8802-2 (ANSI/IEEE Std. 802.2), shall be included in all Messages. The LLC/SNAP header shall be 64 bits (DB16-DB23) and shall immediately follow the MAC header in the first micropacket (Header micropacket). The values of the LLC/SNAP header subfields shall be: DSAP = x'AA', SSAP = x'AA', Ctl = x'03', and the three Org = x'00'. Codings of the EtherType field shall be as assigned in the current "Assigned Numbers" RFC. For the convenience of the reader, HIPPI-6400-specific EtherTypes are listed below:

x'0800' = Non-scheduled IP protocol.

x'8180' = Non-scheduled HIPPI-FP as specified in ANSI X3.210. (See annex A.)

x'8181' = Scheduled Transfer as specified in xxx.

x'8182' = Locally administered.

x'8183' = Reserved

### 7.3 Opaque payload

The eight bytes following the LLC/SNAP header belong to the upper-layer protocol using this Message. The opaque payload bytes may be used to carry additional headers, parameters, or data.

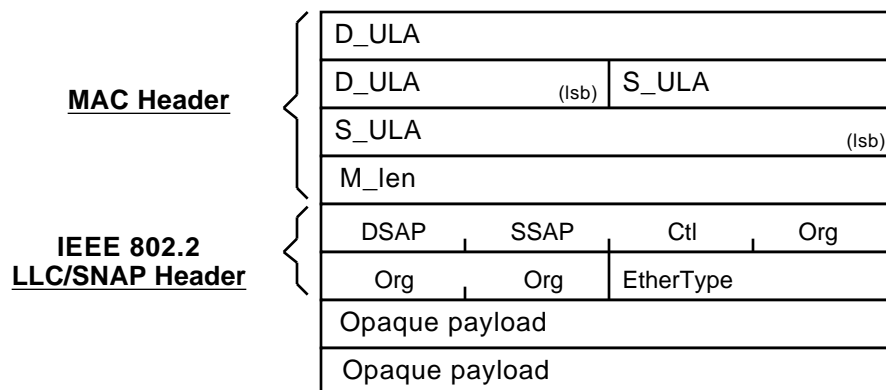


Figure 10 – Header micropacket contents



## 8 Source specific operations

### 8.1 Credit update indications on Source side

Credit update indications from the far end are received on the local Destination side, and passed to the local Source side, as shown in figure 5. A credit update shall increase the available credit, by the amount in the CR parameter, on the virtual channel whose number is the value in the VCR parameter.

If data is ready to be sent on a given VC, but credits are exhausted for this VC (i.e., credit = 0) for the duration of a timeout period, then the link is shut down (see 12.3), and a VC[0-3]\_Credit\_Timeout\_Error logged. The default timeout value shall be 2 seconds (see 13.2).

### 8.2 ACK indications on Source side

ACK indications (see 6.4 and 9.3) from the far end are received on the local Destination side, and passed to the local Source side, as shown in figure 5. An ACK indication acknowledges all of the transmitted micropackets whose TSEQ ≤ RSEQ, i.e., the memory allocated to these micropackets may be re-used. RSEQ = x'FF', which may occur immediately after a Reset operation (see 9.3), shall be ignored.

The ACK indication timeout indicates that a TSEQ was transmitted, but not acknowledged for the length of time longer than the worst-case round trip time possible for an acknowledgment to occur. If the ACK indication timeout expires, the Source shall retransmit all micropackets, (see 8.4), that have not been acknowledged, and shall log an RSEQ\_Missing\_Error (see 13.1). The ACK indication timeout default value shall be 12 μs (see 13.2).

NOTE – The ACK indication timeout provides a recovery mechanism even in the event of lost RSEQ values due to link errors. Faster recovery may be possible with other schemes, e.g., NAKs, but the complexity required for the performance gain did not seem worth it, especially since errors should be infrequent.

If an illegal RSEQ value is received, the Source shall retransmit all micropackets, (see 8.4), that

have not been acknowledged, and log an RSEQ\_Out\_Of\_Range\_Error (see 13.1). An illegal RSEQ is one that does not equal or fall between the last successfully received RSEQ and the highest transmitted but not acknowledged TSEQ.

### 8.3 ACKs and credit updates to far end

The local Destination side sends ACK indications and credit update information to the far end by first queuing them to the local Source side, as shown in figure 5. The Source side shall transmit this information in micropackets using the appropriate control bits. Since the ACK indications and credit update information do not share their fields with any other parameters they can be sent with every micropacket.

The local Destination may queue multiple ACK indication RSEQ parameters before one is transmitted by the local Source end. The RSEQ parameter should be over-written so that the ACK indication Message transmitted uses the latest value of RSEQ.

### 8.4 Micropacket retransmissions

A retransmission sequence, as triggered by 8.2, shall consist of two consecutive training sequences (see 12) followed by retransmission of all of the unacknowledged micropackets in the Output Buffer (see figure 5).

*Open Issue – Preceding the data retransmission with two training sequences is new, and may be removed as ramifications for the HIPPI-6400 Tester are investigated.*

Multiple retransmissions may be required in the event of poor link quality. The link shall be shut down (see 12.3), and a Retransmission\_Error logged (see 13.1), if successful operation is not achieved after a number of successive retransmissions of the same data. The default number is two, and it shall be programmable to other values, including 1 and 4. The mechanisms and procedures used to set values, different from the default value are outside the scope of this standard.

NOTE – This value may need to be larger to accommodate lengthy noise hits.

*Open Issue – The Retransmission\_Error value will be the same as the sum of the RSEQ\_Missing\_Error and RSEQ\_Out\_Of\_Range\_Error. Instead should we count the maximum number of times successive retransmissions occur before achieving success?*

Upon retransmission, the following parameters, from the original micropacket, shall have the same value in the retransmitted micropacket.

- VC
- TYPE
- TAIL
- ERROR
- TSEQ
- VCR
- CR
- ECRC

The following parameters may change as a micropacket is retransmitted.

- RSEQ
- LCRC

## 9 Destination specific operations

### 9.1 Link level processing

The Destination shall process received micropackets in the order of the following subclauses. The unnumbered items within each subclause may be checked in any order. Note that no acknowledgment (i.e., with RSEQ) shall be given for a micropacket that is discarded.

#### 9.1.1 Check received LCRC

- If LCRC syndrome = x'874D' (stomp code) then the Destination shall discard the micropacket, and not log an error.
- If LCRC syndrome  $\neq$  x'0000', and  $\neq$  x'874D' (stomp code), then the Destination shall discard the micropacket and log an LCRC\_Error.

#### 9.1.2 Check received TSEQ

If no errors were detected in 9.1.1, and TSEQ  $\neq$  x'FF', then the following checks shall be made. TYPE  $<$  x'8' is an error. TYPE  $\geq$  x'8', and TSEQ is not one greater than the last non-x'FF', non-stomped, TSEQ received, is also an error. In either case, the micropacket shall be discarded. Additionally, a TSEQ\_Error shall be logged unless no micropackets have been accepted since the last TSEQ\_Error was logged.

#### 9.1.3 Check received ECRC

If no errors were detected in 9.1.1 or 9.1.2, then the following checks shall be made.

- If ERROR = 0 and the ECRC syndrome  $\neq$  x'0000', then the Destination shall discard the micropacket and log an ECRC\_Error.
- If ERROR = 1 and the ECRC syndrome  $\neq$  x'0000', then the Destination shall process the micropacket as if the ECRC were correct (unless this is the Final Destination in which case an error shall be signalled to the ULP).

#### 9.1.4 Unspecified TYPE

If TYPE = unspecified (in the range of x'0' - x'7') then the Destination shall treat the micropacket as a Null micropacket. If TYPE = unspecified (in the range of x'8' - x'F') then intermediate Destinations shall treat the micropacket as a Data micropacket. Treatment by a Final Destination is not specified by this standard. For any unspecified TYPE value, a VC[0-3]\_Undefined\_TYPE\_Error shall be logged and the most recent offending TYPE value stored in Undefined\_TYPE\_Value.

NOTE – The actions applied to Undefined TYPEs are intended to allow for future use of the Undefined TYPEs.

### 9.2 Check for Message protocol errors

Message protocol error checking (at the Destination) shall be done on micropackets that have not been discarded in 9.1 and its subclauses. Since a Message is restricted to a single Virtual Channel, all Message protocol checking shall be applied to each Virtual Channel independently. Credit-only (TYPE = x'A') micropackets shall be ignored for the

purposes of Message protocol checking. Otherwise, micropackets shall be checked in the order received on each Virtual Circuit.

### 9.2.1 Admin missing TAIL bit

If TYPE = Admin, and Tail = 0, then the Destination shall forward the Admin micropacket with ERROR = 1, and TAIL = 1. A VC[1-2]\_Admin\_Tail\_Error shall be logged.

### 9.2.2 Missing start of Message

If a Message is missing the Header micropacket (i.e., a micropacket with TYPE = Data is received following a micropacket with TAIL = 1, or a Link Reset operation) then the Destination shall process the Data micropacket(s) on this VC until a micropacket with TYPE = Header or Admin is received. This processing for the Data micropackets shall consist of discarding the data bytes; their control information shall be treated normally and RSEQs shall be generated. The Header or Admin micropacket shall be treated normally. The Destination shall log a VC[0-3]\_Missing\_Start\_of\_Message\_Error for each discarded Message; not log an error for each discarded Data micropacket.

### 9.2.3 Missing end of Message

If the end of a Message is missing (i.e., TYPE = Header or Admin following a Data, Header, or unspecified TYPE  $\geq$  x'8' micropacket with TAIL = 0) then the Destination shall fabricate an end of Message micropacket (Data Bytes = x'00', VC = as received, TYPE = Data, TAIL = 1, ERROR = 1, other parameters as appropriate). The Destination shall insert the fabricated micropacket into the VC stream, and shall log a VC[0-3]\_Missing\_End\_of\_Message\_Error. The Header or Admin micropacket shall be treated normally.

### 9.2.4 Stall timeout

If a Message is in progress on a VC, that VC's buffer is empty, and no Data micropackets have been received within the Stall timeout period, then the Destination shall fabricate an end of Message micropacket (Data Bytes = x'00', VC = as received, TYPE = Data, TAIL = 1, ERROR = 1, other parameters as appropriate).

The Destination shall insert the fabricated micropacket into the VC stream, and shall log a VC[0-3]\_Stall\_Timeout\_Error. This action flushes the Message in progress. The default value of the Stall timeout shall be 2 ms (see 13.2).

NOTE – Implementors are cautioned that the Stall timeout may be triggered by a slow Source host. If slow hosts are expected, then the Stall timeout value may be set to a larger value to avoid inadvertent actions.

### 9.2.5 No errors detected

If no errors are detected, and TYPE = Header or Data, the micropacket shall be acknowledged and delivered to the virtual channel buffer designated by the VC parameter.

If no errors are detected, and TYPE  $\neq$  Header or Data, the micropacket shall be processed by the Destination.

## 9.3 Generating ACKs

The Destination acknowledges correctly received micropackets by using the RSEQ parameter of micropackets flowing in the reverse direction. Multiple micropackets may be acknowledged with a single RSEQ (e.g., if micropackets with TSEQ = 0,1...7 are received, transmitting RSEQ = 5 acknowledges micropackets 0,1...5, but not 6 and 7). Only micropackets that are not discarded due to errors (see 9.1) and whose TYPE value is in the range x'8' – x'F' shall be acknowledged. If the Destination does not have a new value of RSEQ to send, it shall repeat the last RSEQ value.

Once an error is detected that causes a micropacket not to be acknowledged, the Destination shall not change the RSEQ value until correctly receiving a micropacket with TSEQ = RSEQ + 1 (the retransmission of the micropacket that was in error). Hence, an error will result in a given RSEQ value being continually sent, and the Source timing out waiting for the expected RSEQ value (i.e., RSEQ > last RSEQ).

The Destination shall use RSEQ = x'FF' after a Link Reset or Initialize operation until it has received the micropacket with TSEQ = x'00'.

## 10 Signal line encoding

### 10.1 Signal line bit assignments

The data bytes and control bits shall be transmitted on the signal lines specified in table 3 for a 16-bit wide interface, and as specified in table 4 for an 8-bit wide interface. Nomenclature for the data and control bits is detailed in figure 3. Data signal lines are labeled capital D and a two-digit number, e.g., D00. Control signal lines are labeled capital C and a one-digit number, e.g., C0. The horizontal rows correspond to logical clock ticks. They are grouped in fours, corresponding to the 4b/5b coding.

**Table 3 – Signal line bit assignments in a 16-bit system**

bit	Signal lines																			
	C3	C2	C1	C0	D 15	D 14	D 13	D 12	D 11	D 10	D 09	D 08	D 07	D 06	D 05	D 04	D 03	D 02	D 01	D 00
a	12	08	04	00	07.4	07.0	06.4	06.0	05.4	05.0	04.4	04.0	03.4	03.0	02.4	02.0	01.4	01.0	00.4	00.0
b	13	09	05	01	07.5	07.1	06.5	06.1	05.5	05.1	04.5	04.1	03.5	03.1	02.5	02.1	01.5	01.1	00.5	00.1
c	14	10	06	02	07.6	07.2	06.6	06.2	05.6	05.2	04.6	04.2	03.6	03.2	02.6	02.2	01.6	01.2	00.6	00.2
d	15	11	07	03	07.7	07.3	06.7	06.3	05.7	05.3	04.7	04.3	03.7	03.3	02.7	02.3	01.7	01.3	00.7	00.3
a	28	24	20	16	15.4	15.0	14.4	14.0	13.4	13.0	12.4	12.0	11.4	11.0	10.4	10.0	09.4	09.0	08.4	08.0
b	29	25	21	17	15.5	15.1	14.5	14.1	13.5	13.1	12.5	12.1	11.5	11.1	10.5	10.1	09.5	09.1	08.5	08.1
c	30	26	22	18	15.6	15.2	14.6	14.2	13.6	13.2	12.6	12.2	11.6	11.2	10.6	10.2	09.6	09.2	08.6	08.2
d	31	27	23	19	15.7	15.3	14.7	14.3	13.7	13.3	12.7	12.3	11.7	11.3	10.7	10.3	09.7	09.3	08.7	08.3
a	44	40	36	32	23.4	23.0	22.4	22.0	21.4	21.0	20.4	20.0	19.4	19.0	18.4	18.0	17.4	17.0	16.4	16.0
b	45	41	37	33	23.5	23.1	22.5	22.1	21.5	21.1	20.5	20.1	19.5	19.1	18.5	18.1	17.5	17.1	16.5	16.1
c	46	42	38	34	23.6	23.2	22.6	22.2	21.6	21.2	20.6	20.2	19.6	19.2	18.6	18.2	17.6	17.2	16.6	16.2
d	47	43	39	35	23.7	23.3	22.7	22.3	21.7	21.3	20.7	20.3	19.7	19.3	18.7	18.3	17.7	17.3	16.7	16.3
a	60	56	52	48	31.4	31.0	30.4	30.0	29.4	29.0	28.4	28.0	27.4	27.0	26.4	26.0	25.4	25.0	24.4	24.0
b	61	57	53	49	31.5	31.1	30.5	30.1	29.5	29.1	28.5	28.1	27.5	27.1	26.5	26.1	25.5	25.1	24.5	24.1
c	62	58	54	50	31.6	31.2	30.6	30.2	29.6	29.2	28.6	28.2	27.6	27.2	26.6	26.2	25.6	25.2	24.6	24.2
d	63	59	55	51	31.7	31.3	30.7	30.3	29.7	29.3	28.7	28.3	27.7	27.3	26.7	26.3	25.7	25.3	24.7	24.3
NOTES																				
1 The two-digit numbers in the C <sub>n</sub> columns are the control bits, c <sub>nn</sub> .																				
2 The three-digit numbers in the D <sub>nn</sub> columns are the data bits, d <sub>xx.y</sub> , where xx is the byte number and y is the bit number in the byte.																				
3 The 4-bit groups in a column are transmitted on the associated signal line, top group first, bottom group last.																				
4 The four-bit groups in a column denote 4-bit code groups (dcba) for encoding/decoding to/from the 5-bit transmission codes (zyTxw) specified in table 5. A 5-bit group code (wxTy <sub>z</sub> ) is transmitted over one signal line, e.g., D00.																				

**Table 4 – Signal line bit assignments in an 8-bit system**

bit	Signal lines									
	C1	C0	D 07	D 06	D 05	D 04	D 03	D 02	D 01	D 00
a	08	00	07.0	06.0	05.0	04.0	03.0	02.0	01.0	00.0
b	09	01	07.1	06.1	05.1	04.1	03.1	02.1	01.1	00.1
c	10	02	07.2	06.2	05.2	04.2	03.2	02.2	01.2	00.2
d	11	03	07.3	06.3	05.3	04.3	03.3	02.3	01.3	00.3
a	12	04	07.4	06.4	05.4	04.4	03.4	02.4	01.4	00.4
b	13	05	07.5	06.5	05.5	04.5	03.5	02.5	01.5	00.5
c	14	06	07.6	06.6	05.6	04.6	03.6	02.6	01.6	00.6
d	15	07	07.7	06.7	05.7	04.7	03.7	02.7	01.7	00.7
a	24	16	15.0	14.0	13.0	12.0	11.0	10.0	09.0	08.0
b	25	17	15.1	14.1	13.1	12.1	11.1	10.1	09.1	08.1
c	26	18	15.2	14.2	13.2	12.2	11.2	10.2	09.2	08.2
d	27	19	15.3	14.3	13.3	12.3	11.3	10.3	09.3	08.3
a	28	20	15.4	14.4	13.4	12.4	11.4	10.4	09.4	08.4
b	29	21	15.5	14.5	13.5	12.5	11.5	10.5	09.5	08.5
c	30	22	15.6	14.6	13.6	12.6	11.6	10.6	09.6	08.6
d	31	23	15.7	14.7	13.7	12.7	11.7	10.7	09.7	08.7
a	40	32	23.0	22.0	21.0	20.0	19.0	18.0	17.0	16.0
b	41	33	23.1	22.1	21.1	20.1	19.1	18.1	17.1	16.1
c	42	34	23.2	22.2	21.2	20.2	19.2	18.2	17.2	16.2
d	43	35	23.3	22.3	21.3	20.3	19.3	18.3	17.3	16.3
a	44	36	23.4	22.4	21.4	20.4	19.4	18.4	17.4	16.4
b	45	37	23.5	22.5	21.5	20.5	19.5	18.5	17.5	16.5
c	46	38	23.6	22.6	21.6	20.6	19.6	18.6	17.6	16.6
d	47	39	23.7	22.7	21.7	20.7	19.7	18.7	17.7	16.7
a	56	48	31.0	30.0	29.0	28.0	27.0	26.0	25.0	24.0
b	57	49	31.1	30.1	29.1	28.1	27.1	26.1	25.1	24.1
c	58	50	31.2	30.2	29.2	28.2	27.2	26.2	25.2	24.2
d	59	51	31.3	30.3	29.3	28.3	27.3	26.3	25.3	24.3
a	60	52	31.4	30.4	29.4	28.4	27.4	26.4	25.4	24.4
b	61	53	31.5	30.5	29.5	28.5	27.5	26.5	25.5	24.5
c	62	54	31.6	30.6	29.6	28.6	27.6	26.6	25.6	24.6
d	63	55	31.7	30.7	29.7	28.7	27.7	26.7	25.7	24.7
NOTES:										
1 The two-digit numbers in the Cn columns are the control bits, cnn.										
2 The three-digit numbers in the Dnn columns are the data bits, dxx.y, where xx is the byte number and y is the bit number in the byte.										
3 The 4-bit groups in a column are transmitted on the associated signal line, top group first, bottom group last.										
4 The four-bit groups in a column denote 4-bit code groups (dcba) for encoding/decoding to/from the 5-bit transmission codes (zyTxw) specified in table 5. A 5-bit code group (wxTyx) is transmitted over one signal line, e.g., D00.										

## 10.2 Source-side encoding for dc balance

The transmitted signals shall be encoded to achieve dc balance on each signal line. Table 5 specifies the 5-bit signal line codes (zyTxw) corresponding to the 4-bit input codes (dcba) from tables 3 and 4. For example, on signal line D00, the first dcba 4-bit code consists of bits d00.0, d00.1, d00.2, and d00.3. See annex A.1 for an example circuit.

A running count, called the Disparity Count, shall be kept of all the ones and zeros transmitted since the link was reset. The Disparity Count shall be incremented for each one transmitted, and decremented for each zero transmitted.

The appropriate 5-bit code value from table 5, based on the current value of the Disparity Count, shall be transmitted in the sequence, w,x,T,y,z. (T = true/complement bit) For example in the right column of tables 3 and 4, if:

a = d00.0 = 1 (least-significant bit)

b = d00.1 = 0

c = d00.2 = 0

d = d00.3 = 0

and Disparity Count = +1 before encoding, then, based on the third column second row in table 5, transmit on D00:

w = 1 (transmitted first)

x = 0

T = 1

y = 0

z = 0

Disparity Count = 0 after encoding.

### NOTES

1 The range for the Disparity Count at the 5-bit boundaries is from +4 to -5. The range for the Disparity Count is from +6 to -7.

2 The Disparity Count may also be updated by adding or subtracting the value of Delta Disparity shown in table 5. Add Delta Disparity if Disparity Count < 0; subtract if ≥ 0.

3 The 5-bit code is derived by inserting a 1 in the middle of the 4-bit code, and then transmitting either the true or complement value of the resultant 5-bit quantity.

4. The maximum run length, i.e., the longest string of continuous 1s or 0s, is 11. The string of 4-bit code points creating the maximum run length is 'EFC'. Start with Disparity Count = +3 or +4 for a

string of 11 zeros. Start with Disparity Count = -4 or -5 for a string of 11 ones.

The data and control signal lines shall be synchronized with the CLOCK, CLOCK\_2, and FRAME signals as shown in figures 10 and 12. Figures 11 through 14 are read left to right, i.e., events on the left occur before those on the right. In figures 11 and 13, the CLOCK\_2 signal is deliberately shown skewed in relation to the CLOCK signal, although in actual implementation it may not be skewed (see 14.1).

**Table 5 – 4b/5b line coding**

4-bit code dcba	5-bit code when Disparity < 0 zyTxw	5-bit code when Disparity ≥ 0 zyTxw	Delta Disparity
0000	11011	00100	3
0001	11010	00101	1
0010	11001	00110	1
0011	00111	11000	1
0100	10011	01100	1
0101	01101	10010	1
0110	01110	10001	1
0111	01111	10000	3
1000	01011	10100	1
1001	10101	01010	1
1010	10110	01001	1
1011	10111	01000	3
1100	11100	00011	1
1101	11101	00010	3
1110	11110	00001	3
1111	11111	00000	5

## 10.3 Destination-side decoding

The received signals shall each be decoded in groups of five bits according to table 5.

### NOTES

1 Decoding can be implemented by examining the middle bit of the 5-bit code; if 1 then use the outer bits uncomplemented, if 0 then complement before use.

2 There are no illegal 5-bit codes.

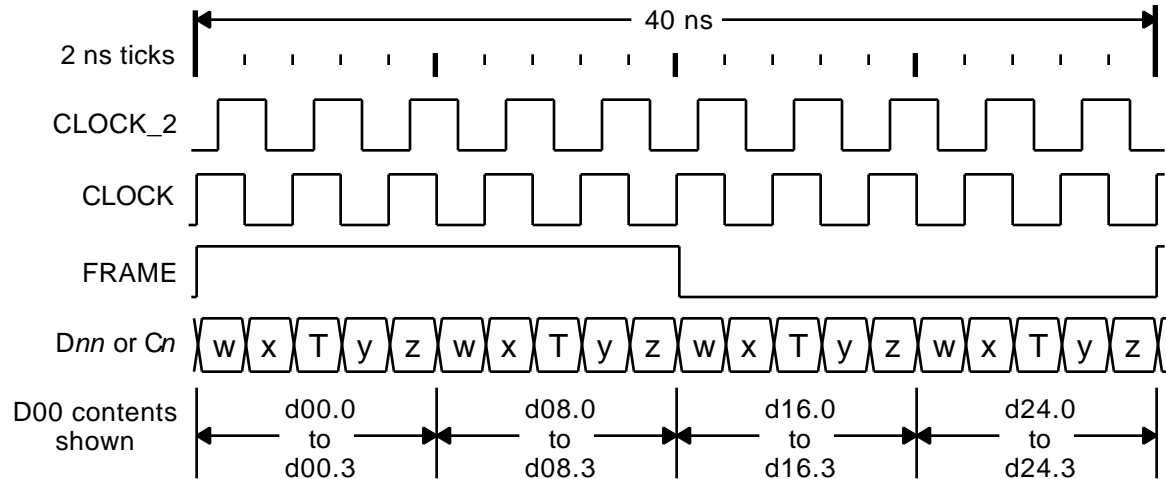


Figure 11 – 16-bit system micropacket

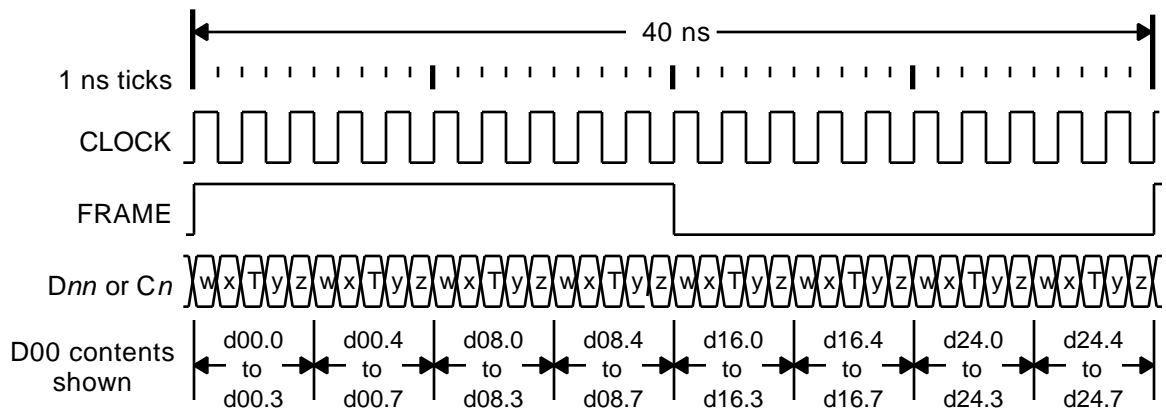


Figure 12 – 8-bit system micropacket

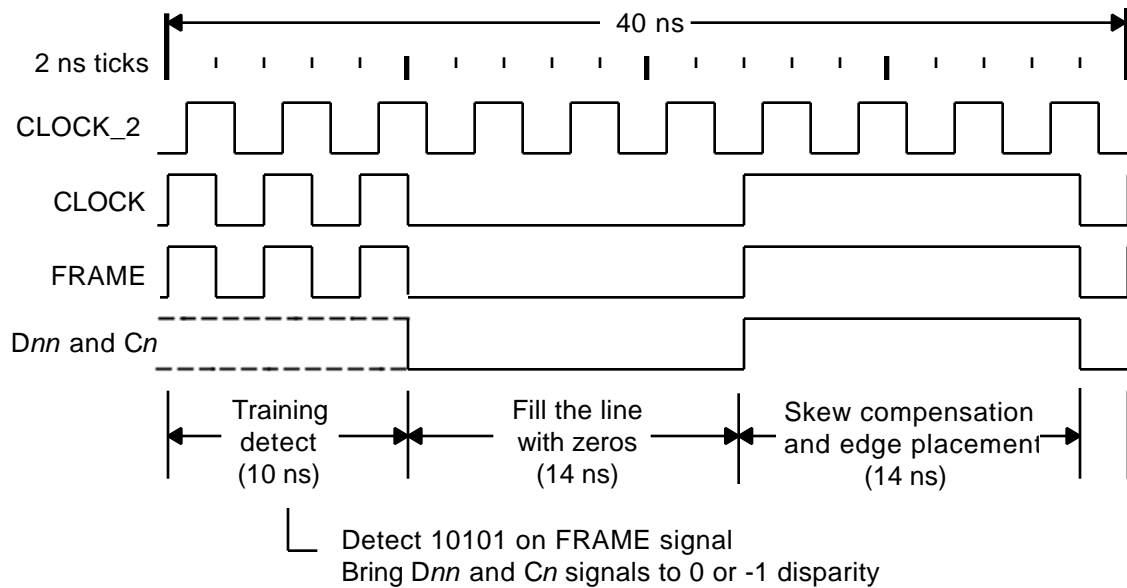
#### 10.4 FRAME signal

The FRAME signal transitions shall be as shown in figures 11 through 14. As shown in figures 13 and 14, the start of a training sequence (40 ns long) shall be signaled by a 10101 FRAME signal pattern.

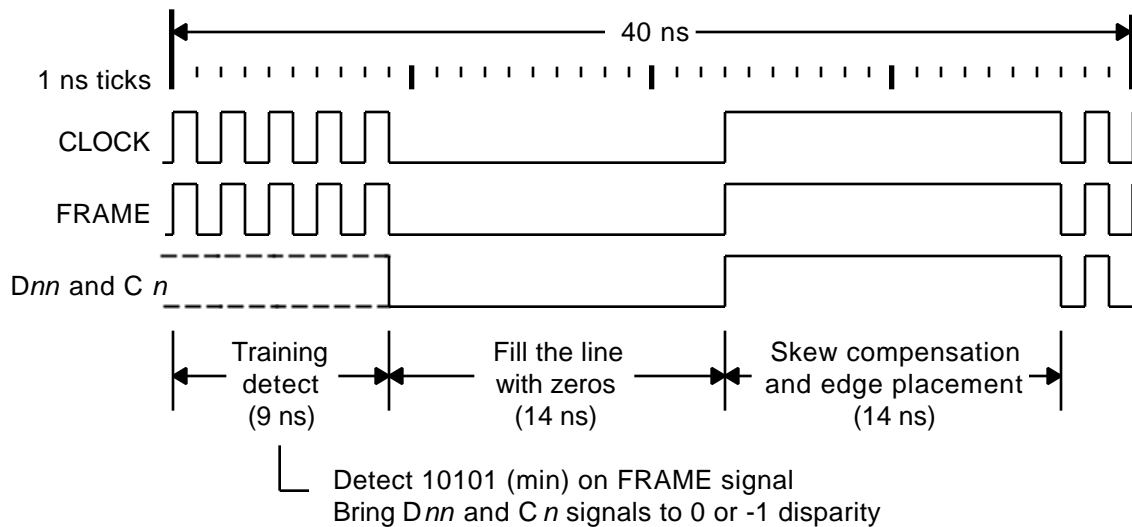
As shown in figures 11 and 12, a 0 to 1 transition on the FRAME signal shall signal the beginning of a micropacket, unless the transition is part of a training sequence. In a micropacket, the FRAME signal shall remain = 1 for the first half of the micropacket (20 ns), and shall = 0 for the last half (20 ns).

#### 11 Link training

The Destination shall compensate for up to 10 ns of skew among the signals. Skew is defined as the time between the earliest and latest signal arrival at the Destination. Training sequences (see figures 13 and 14) shall be used to measure the skew, and perform dynamic skew adjustments. A FRAME signal pattern of 10101, as specified in 10.4, shall be used to identify a training sequence.



**Figure 13 – 16-bit system training sequence**



**Figure 14 – 8-bit system training sequence**

*Open Issue –The training sequence has changed, hopefully for the last time. It needs to be checked for correctness and clarity.*



A single training sequence shall be inserted by the Source at least every 10  $\mu$ s to adjust the dynamic skew, and also to compensate for CLOCK frequency differences between the Source and Destination (see annex A.2). During the first portion of a training sequence the Source shall insert appropriate Data and Control bits to drive the Disparity Count (see 10.2) on those signal lines to 0 or -1. The Disparity Count shall be set to zero at the end of the training sequence.

If a training sequence is unsuccessful, after the link had been healthy, then a Skew\_Retraining\_Error shall be logged.

## 12 Initialization and Shutdown

Two levels of initialization are specified, Link Reset and Initialize. They differ in whether the action may be propagated to other links. These operations are diagrammed in figure 15. Link Shutdown occurs when the link is fatally flawed, and requires administrator intervention for recovery.

### 12.1 Link Reset

Link Reset affects the local link only, it is not propagated to other links. Link Reset may be triggered by power-on, the local administrator, a Port\_Reset Admin command, or receiving a micropacket with TYPE = Reset.

A Link Reset operation shall execute training sequences, and reset the local state, exiting with:

- All of the VC input and output buffers shall be emptied.
- Credit for all of the VCs shall be set to zero.
- TSEQ shall be reset to x'00'; RSEQ shall be set to x'FF'.
- The dynamic skew compensation circuitry adjusted, and micropackets being received correctly.
- The Disparity Count will have been set to minus one.

- The other end of the link will have also executed a Link Reset operation.

During a Link Reset operation micropackets with TYPE  $\neq$  x'2' - x'5' shall be discarded, and the error logging specified in clause 9 shall not occur. The error counts accessible by Admin operation (see table 6) shall not be modified by a Link Reset operation except when first coming out of a power-on or Initialize operation.

### 12.2 Initialize

Initialize resets the local link as defined in 12.1, and is passed to the administrator for possible propagation to other entities. Initialize may be triggered by the local administrator, a Port\_Initialize Admin command, or receiving a micropacket with TYPE = Initialize (see 6.3.1).

During an Initialize operation micropackets with TYPE  $\neq$  x'4' or x'5' shall be discarded, and the error logging specified in clause 9 shall occur. The error counts accessible by Admin operations (see table 6) shall not be modified by an Initialize operation. All timeout timers (see table 7) shall be initialized at the completion of the Initialize operation.

### 12.3 Link Shutdown

A Link Shutdown occurs when:

- The number of times retransmission occurs exceeds some limit (see 8.2).
- The Source has been unable to transmit due to lack of credit (see 8.1).
- A Destination receives micropackets for a VC whose VC Buffer (see figure 5) is full. In this case, a VC[0-3]\_RX\_VC\_Buffer\_Overflow error logged shall be logged.
- The link does not come out of a Link Reset or Initialize operation within a default time of 2 seconds (see 13.2). In this case, a Skew\_Training\_Reset\_Error shall also be logged.

*Open Issue – Should credit overflow cause a Shutdown? Link Reset? Should an error be logged?*

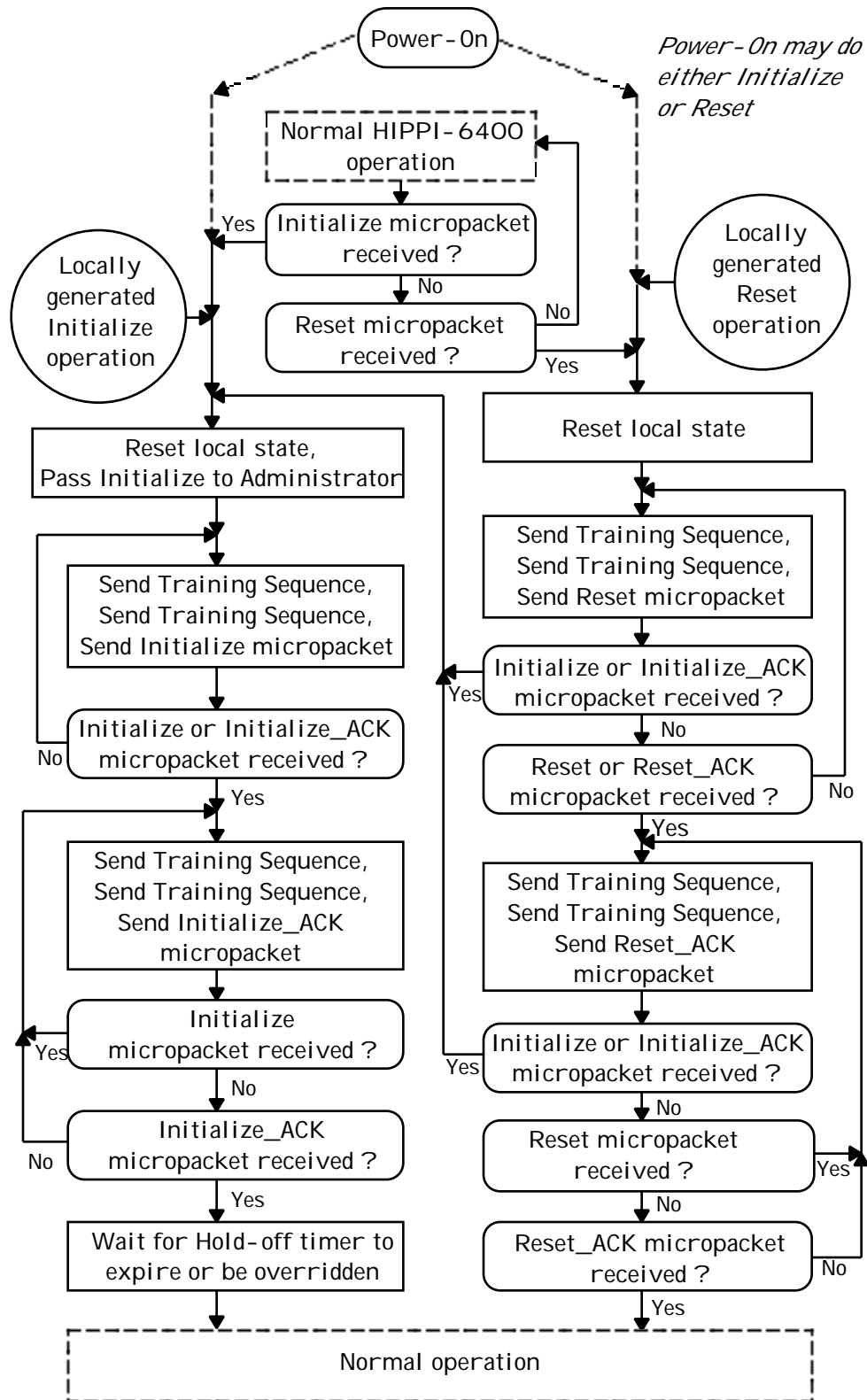


Figure 15 – Initialize and Link Reset operations

All of the VC input and output buffers shall be emptied. While the link is shutdown, incoming micropackets with TYPE = x'2'-x'5' shall be processed, all other micropackets shall be discarded. External administrative action beyond the scope of this standard may be required to recover from a Link Shutdown. Note that administrative actions may clear the error counts accessible by Admin operations (see table 6).

Until expired, the Hold-off timer shall be used to prohibit incoming TYPE = Initialize micropackets from starting an Initialize sequence. The Hold-off timer shall be started by receipt of a TYPE = Initialize\_ACK micropacket. The default value for the Hold-off timer shall be 5 seconds (see 13.2). The intent of the Hold-off timer is to prevent infinite Initialize oscillations between connected devices.

## 13 Maintenance and control features

### 13.1 Logged errors

Table 6 contains a summary of the errors that shall be logged, the minimum number of bits for the parameter, and the location in this standard discussing the error. A counter shall not roll over if its maximum value is reached.

### 13.2 Timeouts

Table 7 contains a summary of the timeouts, their default value, and the location in this standard discussing the timeout. All of the timeouts shall be programmable, at least to values of 2X, 1/2X, and 1/4X. The mechanisms and procedures used to set values, different from the default values, are outside the scope of this standard.

*Open Issue - Do we want to standardize all possible maintenance and control Messages?*

*Open Issue - What is the relative priority of maintenance and control Messages?*

*Open Issue - Should we have a MIB for a HIPPI-6400-PH link? For other HIPPI-6400-PH? Who is going to write it?*

## 14 Timing

### 14.1 Source CLOCK signals

The transmitted CLOCK, and CLOCK\_2, signals at the Source bulkhead connector shall have a nominal period of 4 ns  $\pm$  0.4 ps ( $\pm$  0.01%, i.e., 100 parts per million) for a 16-bit system, and a period of 2 ns  $\pm$  0.2 ps ( $\pm$  0.01%, i.e., 100 parts per million) for an 8-bit system. During a training sequence, the Source CLOCK signal waveform shall be as shown in figures 13 and 14. The CLOCK\_2 signal shall be a constant square wave. Note that the CLOCK\_2 signal is only present in 16-bit systems. The phase relationship between CLOCK and CLOCK\_2 shall be any constant value.

*Open Issue – The CLOCK\_2 frequency of 250 MHz is open for discussion – it may be changed to a lower frequency for easier detection.*

CLOCK, and CLOCK\_2, symmetry, measured as the percentage of time in the high state compared to the total CLOCK period, shall be 50 ( $\pm$ 5) %. Peak jitter shall be less than 0.05 ns for a 16-bit system; less than 0.025 ns for an 8-bit system.

*Open Issue – The CLOCK tolerance was changed from 200 ppm to 100 ppm. Does this cause anyone problems?*

### 14.2 Destination CLOCK signals

The intended uses of the separate CLOCK and CLOCK\_2 signals are:

- to provide a separate signal that can be monitored for activity without affecting the signal used for strobing the other signals. If inactivity is detected, the other signals should be ignored to avoid spurious error indications.
- to support different skew compensation implementations. The CLOCK signal used to strobe the signals during retraining shall also be used to strobe the signals during regular operation.

**Table 6 – Summary of logged errors**

Name	Minimum Number of bits	Reference
ECRC_Error	8	9.1.3
ECRC_Source_Error	8	6.6.3
LCRC_Error	8	9.1.1
Retransmission_Error	8	8.4
RSEQ_Missing_Error	8	8.2
RSEQ_Out_Of_Range_Error	8	8.2
Skew_Retraining_Error	1	12
Skew_Training_Reset_Error	1	12.3
TSEQ_Error	8	9.1.2
Undefined_TYPE_Value	4	9.1.4
VC[1-2]_Admin_Tail_Error	1/2	9.2.1
VC[0-3]_Missing_End_of_Message_Error	1/4	9.2.3
VC[0-3]_Missing_Start_of_Message_Error	1/4	9.2.2
VC[0-3]_Stall_Timeout_Error	1/4	9.2.4
VC[0-3]_RX_VC_Buffer_Overflow	1/4	12.3
VC[0-3]_Undefined_TYPE_Error	1/4	9.1.4
VC[0-3]_Credit_Timeout_Error	1/4	8.1
NOTE – The 1/4 entries under the Number of bits column means that there is one bit for an error, e.g., for VC0_Missing_End_of_Message_Error, and a total of four errors possible (i.e., one for each VC).		

**Table 7 – Summary of timeouts**

Name	Default value	Reference
ACK indication timeout	12 $\mu$ s	8.2
Credit timeout	2 s	8.1
Hold-off timer	5 s	12.3
Reset or Initialize timeout	2 s	12.3
Stall timeout	2 ms	9.2.4

### 14.3 FRAME, Data, and Control signals

The FRAME signal shall be strobed on the rising edge of the CLOCK signal. The Data and Control signals shall be strobed on both edges of the CLOCK signal.

*Open Issue – These values were picked as the voltages at the SuMAC, and will need to be adjusted for measurement at the bulkhead connector.*

The output coupling network shall consist of ....

*Open Issue – What should the Reserved pins be tied to? Signal ground? Shield ground? Left open?*

## 15 Copper interface (optional)

### 15.1 General

The copper interface is only applicable to 16-bit systems. Unless otherwise specified, all parameters shall be measured at the bulkhead connector of the Source or Destination equipment. Specifications shall be met when operating with a specified cable and termination. Figure 16 shows the components used in a signal path.

### 15.2 Electrical output interface

Differential drivers shall be used on all signal lines. Signals in the 'true' or '1' state shall have the xx\_Out\_p pin more positive than the xx\_Out\_n pin. Rise and fall times shall be measured at the 20% and 80% points of the signal transition.

- Maximum high-level output voltage = 2.69 V
- Minimum high-level output voltage = 2.20 V
- Maximum low-level output voltage = 0.28 V
- Minimum low-level output voltage = 0.08 V
- Maximum rise time = 300 ps
- Minimum rise time = 100 ps
- Maximum fall time = 300 ps
- Minimum fall time = 100 ps

### 15.3 Electrical input interface

All differential line receivers shall operate correctly when receiving signals meeting the following voltage specifications at the Destination connector:

- Minimum peak-to-peak differential input voltage = 150 mV
- Input High voltage  $\leq$
- Input Low voltage  $\geq$

The coupling network between the cable and the receiver shall consist of ....

*Open Issue – The parameter values need to be filled in, and the coupling network specified.*

### 15.4 Electrical connector

The cable connectors shall be a two-row, 100-pin connectors as shown in figure xx. Pin assignments are shown in figure 17. These specifications shall apply for up to 10,000 mating cycles. Each pin shall have a  $\geq 1$  A current capability, with the total current capability for all pins simultaneously shall be  $\geq 5$  A.

*Open Issue – The connector drawings will be added when time permits.*

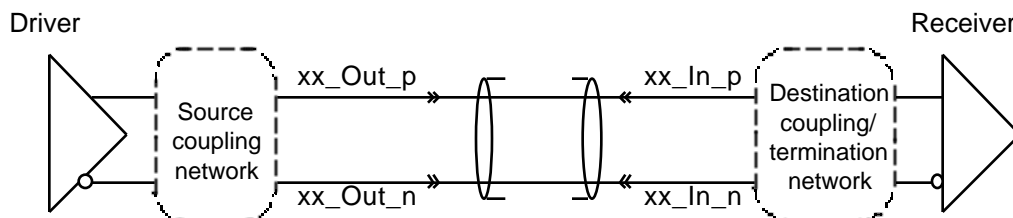


Figure 16 – Signal path

*Open Issue – The connector pin assignments shown in figure 17 changed since Rev 0.5. The new assignments need to be checked.*

The connectors shall provide RFI/EMI shielding sufficient to pass all appropriate compliance tests.

Attenuation shall be  $\leq 0.1$  dB. When multiple pairs are driven differentially with a 500 ps risetime (10% – 90%) pulse, near end crosstalk shall be  $\leq 10\%$ .

Latches shall be used to hold the connectors in the mated position. To prevent damage, if someone trips over a cable, the connector shall un-mate, without damage, when a pull force of ??? pounds is applied in the direction of the cable exit.

*Open Issue – This is a first pass at these specifications, hence they need review and refinement.*

## 15.5 Cable specifications

The cable shall provide differential paths for 46 signals, 23 in each direction. Cable length is determined by cable quality, environmental factors, and the possible use of equalizers. All cable assemblies (cable and optional equalizer if used) shall meet the following specifications:

- Impedance =  $150\ \Omega \pm 10\%$
- Pair-to-pair skew  $\leq 5$  ns
- Within-pair skew  $\leq 250$  ps
- Attenuation @ 250 MHz  $\leq 20$  dB
- Outside diameter =  $0.65$  in  $\pm 0.015$  in ( $16.5$  mm  $\pm 0.38$  mm)
- Jacket material = CL-2.2/FT6 (plenum rated)
- Bend radius  $\leq 6$  in ( $152$  mm)

The cable shall provide individual shields, or equivalent, for each differential path. These individual shields shall be floating, i.e., not connected to each other, to the overall shield, or at the connector. There shall be an overall shield that shall be grounded by .....

## 16 Optical interface (optional)

### 16.1 General

*Open Issue - Width of interface – 8+2+1+1?*

*Open Issue - Signalling frequency, tolerance*

*Open Issue - What is the shape of the FRAME signal?*

### 16.2 Optical output interface

*Open Issue - Optical parameters – wavelength, power, ...*

*Open Issue - What is the output skew specification?*

### 16.3 Optical input interface

*Open Issue - Optical parameters – power...*

*Open Issue - What is the input skew specification?*

### 16.4 Optical connector

*Open Issue - Connector pin assignments*

*Open Issue - Connector specifications?*

*Open Issue - Different connectors at Source and Destination? Same with keying?*

*Open Issue - Field termination of connectors*

### 16.5 Optical cable specifications

*Open Issue - Cable type, number of fibers*

*Open Issue - Cable length*

*Open Issue - Cable optical specs. – loss, crosstalk, relative skew, ...*

*Open Issue - Cable mechanical – fiber pitch, size, jacket material*

**xx Other Open Issues**

*Open Issue - Should we have an annex describing the use of HIPPI-6400-PH links in networks? It could show a group of links crossing a switch and illustrate how the VC is unchanged through the fabric, and most of the other parameters are local to the links.*

Shield	51	1	Reserved
Reserved	52	2	CLOCK_Out_p
CLOCK_Out_p	53	3	CLOCK_Out_n
CLOCK_Out_n	54	4	Reserved
Reserved	55	5	FRAME_In_p
FRAME_Out_p	56	6	FRAME_In_n
FRAME_Out_n	57	7	Reserved
Reserved	58	8	D08_Out_p
D00_Out_p	59	9	D08_Out_n
D00_Out_n	60	10	D09_Out_p
D01_Out_p	61	11	D09_Out_n
D01_Out_n	62	12	D10_Out_p
D02_Out_p	63	13	D10_Out_n
D02_Out_n	64	14	D11_Out_p
D03_Out_p	65	15	D11_Out_n
D03_Out_n	66	16	D12_Out_p
D04_Out_p	67	17	D12_Out_n
D04_Out_n	68	18	D13_Out_p
D05_Out_p	69	19	D13_Out_n
D05_Out_n	70	20	D14_Out_p
D06_Out_p	71	21	D14_Out_n
D06_Out_n	72	22	D15_Out_p
D07_Out_p	73	23	D15_Out_n
D07_Out_n	74	24	D08_In_p
D00_In_p	75	25	D08_In_n
D00_In_n	76	26	D09_In_p
D01_In_p	77	27	D09_In_n
D01_In_n	78	28	D10_In_p
D02_In_p	79	29	D10_In_n
D02_In_n	80	30	D11_In_p
D03_In_p	81	31	D11_In_n
D03_In_n	82	32	D12_In_p
D04_In_p	83	33	D12_In_n
D04_In_n	84	34	D13_In_p
D05_In_p	85	35	D13_In_n
D05_In_n	86	36	D14_In_p
D06_In_p	87	37	D14_In_n
D06_In_n	88	38	D15_In_p
D07_In_p	89	39	D15_In_n
D07_In_n	90	40	C0_In_p
C0_Out_p	91	41	C0_In_n
C0_Out_n	92	42	C1_In_p
C1_Out_p	93	43	C1_In_n
C1_Out_n	94	44	C2_In_p
C2_Out_p	95	45	C2_In_n
C2_Out_n	96	46	C3_In_p
C3_Out_p	97	47	C3_In_n
C3_Out_n	98	48	CLOCK_2_In_p
CLOCK_2_Out_p	99	49	CLOCK_2_In_n
CLOCK_2_Out_n	100	50	Shield

**Figure 17 – Bulkhead connector pin assignments**

## Annex A (informative)

### Implementation comments

#### A.1 4b/5b encoding and decoding

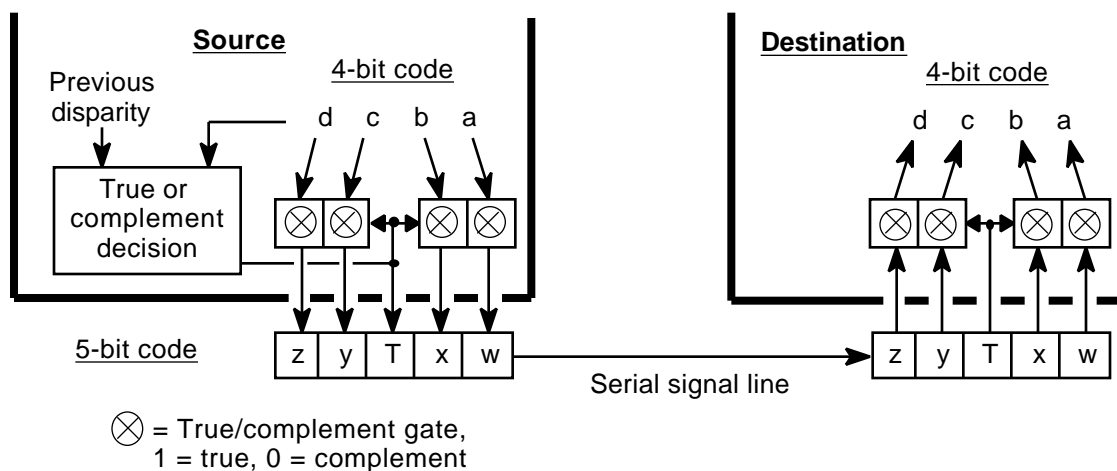
Encoding the 4-bit code groups into 5-bit transmission codes may be implemented as shown in the left portion of the example in figure A.1. Decoding the 4-bit code from the 5-bit code may be implemented as shown in the right portion of figure A.1. The specification for the encoding and decoding is in 10.2 and 10.3.

#### A.2 Frequency differences between Source and Destination

Although the two ends of a HIPPI-6400 link run at nominally the same speed, there can be very slight differences in clock frequency due to inaccuracy of the crystal oscillators at each end. If a transmitter is allowed to send an very long burst of continuous traffic, this will eventually

cause a receiver to overrun if that receiver's clock is slightly slower than the transmitter's clock.

To prevent this condition, the length of continuous data transmission is limited by inserting non-data micropackets (training sequences in HIPPI-6400-PH) periodically. The frequency of training sequences is determined by the potential inaccuracy of the oscillators and the amount of drift the receiver can tolerate. With  $\pm 100$  ppm of frequency error (see 14.1), the total clock error could be as large as 200 ppm, since the sender and receiver could be off in opposite directions. Allowing a drift of 4 ns before correction, takes  $4 \text{ ns} \times (1/200 \text{ ppm}) = 20 \text{ } \mu\text{s}$ . Hence, the requirement that HIPPI-6400-PH transmitters insert retraining sequences at least every 10  $\mu\text{s}$ .



**Figure A.1 – Encode / decode circuit example**



### A.3 LCRC parallel implementation

The LCRC specified in 6.6.2 and figure 8 is based on a bit-by-bit serial implementation. Parallel implementations may be used as long as they produce the same results as the serial example. Tables A.3 and A.4 give equations for 16-bit and 64-bit parallel LCRC implementations, useful for LCRC generation as shown in figure A.2. Table A.5 gives 80-bit parallel equations for LCRC checking, as shown in figure A.3. Other parallel widths may be used, these are just examples.

For these LCRC equations, c63 through c48 are the flip-flops shown in figure 8, and the resultant LCRC control bits. *bn* are the bits which must be delivered to the parallel equation simultaneously. b0 is the first bit which would have been supplied to the serial implementation. The *rn* bits are the all 1's seed value, and the intermediate results from the Partial LCRC Register.

#### A.3.1 Parallel LCRC generator

Parallel LCRC generation can be accomplished by cascading 16-bit parallel equations and 64-bit parallel equations as shown in Figure A.2. Four clock periods are used to produce the LCRC value for a micropacket. Table A.1 summarizes the input bits for each clock period.

**Table A.1 – Parallel LCRC input bits**

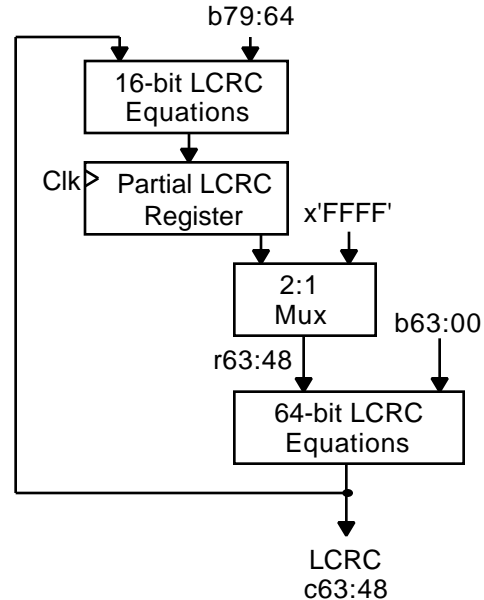
Clock Period	b79:64	b63:00	Mux output
1	c00–c15	d00.0–d07.7	x'FFFF'
2	c16–c31	d08.0–d15.7	partial
3	c32–c47	d16.0–d23.7	partial
4	c48–c63	d24.0–d31.7	partial

a During the first period, the c00–c15 are applied to the 16-bit equations, and the 64 bits d00.0–d07.7 are applied to the 64-bit LCRC equations. Note that for this first cycle only, the multiplexer is set to force x'FFFF' as the 16-bit partial LCRC value, (i.e., initializing with a value of all ones). The register is clocked after the signals have settled.

b During the second period, c16–c31 are applied to the 16-bit equations, and d08.0–d15.7 are applied to 64-bit equations. The register is clocked a second time.

c During the third period, c32–c47 are applied to the 16-bit equations, and d16.0–d23.7 are applied to the 64-bit equations. The register is clocked a third time.

d During the fourth, and final, period, the c48–c63 values presented to the 16-bit equations are immaterial (they are just included for consistency with the LCRC checker), and d24.0–d31.7 are applied to the 64-bit equations – the register is not clocked. After appropriate settling time, the LCRC is available as c63–c48 (c63 is the msb).

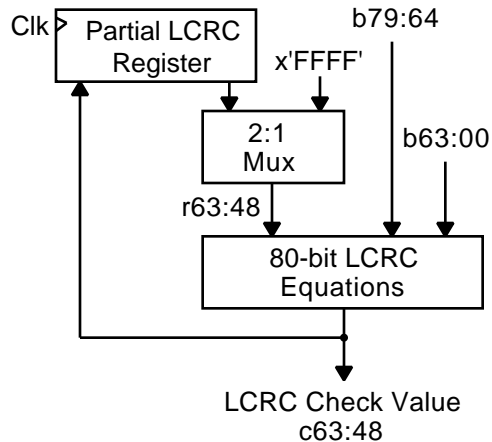


**Figure A.2 – Parallel LCRC generator example**

#### A.3.2 Parallel LCRC checker

Like the LCRC generator, the LCRC checker uses four clock periods to produce the LCRC check value. The LCRC checker can use a single set of 80-bit equations as shown in figure A.3 rather than cascading 16-bit and 64-bit equations. The difference between the generator and checker is that the generator does not include the LCRC bits (c63:48) in the calculation's final step, while the checker

includes them. A final LCRC value of x'0000' means no error; x'06A9' means a stomp code. The input bits are also summarized in table A.1, and the time steps are essentially the same.



**Figure A.3 – Parallel LCRC checker example**

#### A.4 ECRC parallel implementation

The ECRC specified in 6.6.3 and figure 9 is based on a bit-by-bit serial implementation. Parallel implementations may be used as long as they produce the same results as the serial example. Table A.6 gives equations for a 64-bit parallel ECRC implementation as shown in figure A.4. Other parallel widths may be used, this is just an example.

For these ECRC equations, c47 through c32 are the flip-flops shown in figure 9, and the resultant ECRC control bits.  $bn$  are the bits which must be delivered to the parallel equation simultaneously.  $b0$  is the first bit which would have been supplied to the serial implementation. The  $rn$  bits are the all 1's seed value, and the intermediate results from the Partial ECRC Register. Four partial ECRC registers are required since the ECRC is continued across multiple micropackets, and the micropackets from different VC's can be interleaved. Four clock periods are used to

produce the ECRC value for a micropacket. Table A.2 summarizes the input bits for each clock period.

**Table A.2 – Parallel ECRC input bits**

Clock Period	b63:00	Mux output
1	d00.0–d07.7	(see text)
2	d08.0–d15.7	partial
3	d16.0–d23.7	partial
4	d24.0–d31.7	partial

a During the first period, the 64 bits d00.0–d07.7 are applied to the 64-bit ECRC equations. Note that for this first cycle only, and only if this is the first micropacket of a Message, the multiplexer is set to force x'FFFF' as the 16-bit partial ECRC value, (i.e., initializing with a value of all ones). The appropriate VC partial register is clocked after the signals have settled.

b During the second period, d08.0–d15.7 are applied to 64-bit equations. The appropriate register is clocked a second time.

c During the third period, d16.0–d23.7 are applied to the 64-bit equations. The appropriate register is clocked a third time.

d During the fourth, and final, period, d24.0–d31.7 are applied to the 64-bit equations. After appropriate settling time, and without clocking the register, the ECRC is available as c63–c48 (c63 is the msb).

#### A.5 Undetected errors

Simulations have shown that all cases of up to five simultaneous bit errors in a micropacket are detected. Four cases of 4-bit errors are not detected by LCRC or ECRC errors, but are detected by other tests, e.g., bad TSEQ values.

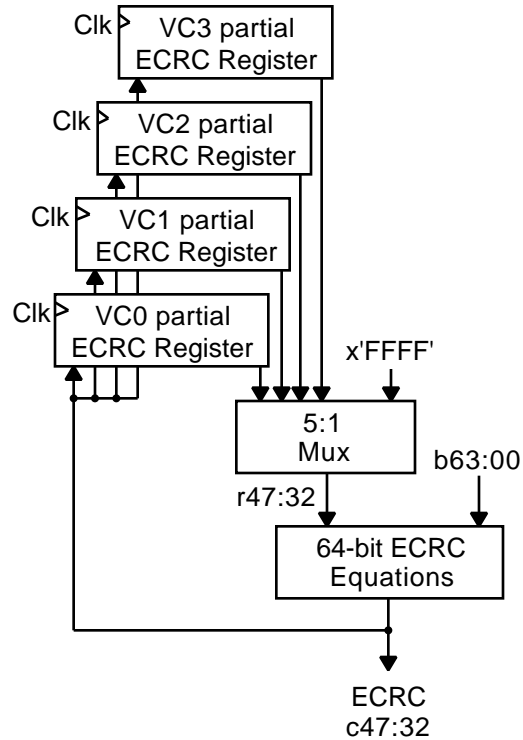


Figure A.4 – Parallel ECRC example

Table A.3 – 16-bit LCRC generator equations

Output	Exclusive OR these bits together															
r63	b79	b75	b71	b68	b67	c63	c59	c55	c52	c51						
r62	b78	b74	b70	b67	b66	c62	c58	c54	c51	c50						
r61	b77	b73	b69	b66	b65	c61	c57	c53	c50	c49						
r60	b76	b72	b68	b65	b64	c60	c56	c52	c49	c48						
r59	b75	b71	b67	b64	c59	c55	c51	c48								
r58	b79	b75	b74	b71	b70	b68	b67	b66	c63	c59	c58	c55	c54	c52	c51	c50
r57	b78	b74	b73	b70	b69	b67	b66	b65	c62	c58	c57	c54	c53	c51	c50	c49
r56	b77	b73	b72	b69	b68	b66	b65	b64	c61	c57	c56	c53	c52	c50	c49	c48
r55	b76	b72	b71	b68	b67	b65	b64	c60	c56	c55	c52	c51	c49	c48		
r54	b75	b71	b70	b67	b66	b64	c59	c55	c54	c51	c50	c48				
r53	b74	b70	b69	b66	b65	c58	c54	c53	c50	c49						
r52	b73	b69	b68	b65	b64	c57	c53	c52	c49	c48						
r51	b79	b75	b72	b71	b64	c63	c59	c56	c55	c48						
r50	b78	b74	b71	b70	c62	c58	c55	c54								
r49	b77	b73	b70	b69	c61	c57	c54	c53								
r48	b76	b72	b69	b68	c60	c56	c53	c52								

**Table A.4 – 64-bit LCRC generator equations**

<b>Output</b>	<b>Exclusive OR these bits together</b>															
c63	b63 b14	b59 b12	b55 b11	b52 b8	b51 b7	b44 b5	b43 b0	b41 r63	b37 r62	b36 r60	b35 r59	b31 r56	b30 r55	b28 r53	b21 r48	b15
c62	b62 b13	b58 b11	b54 b10	b51 b7	b50 b6	b43 b4	b42 r62	b40 r61	b36 r59	b35 r58	b34 r55	b30 r54	b29 r52	b27	b20	b14
c61	b61 b12	b57 b10	b53 b9	b50 b6	b49 b5	b42 b3	b41 r61	b39 r60	b35 r58	b34 r57	b33 r54	b29 r53	b28 r51	b26	b19	b13
c60	b60 b11	b56 b9	b52 b8	b49 b5	b48 b4	b41 b2	b40 r60	b38 r59	b34 r57	b33 r56	b32 r53	b28 r52	b27 r50	b25	b18	b12
c59	b59 b10	b55 b8	b51 b7	b48 b4	b47 b3	b40 b1	b39 r59	b37 r58	b33 r56	b32 r55	b31 r52	b27 r51	b26 r49	b24	b17	b11
c58	b63 b35 b5	b59 b32 b3	b58 b28 b2	b55 b26 r63	b54 b25 r62	b52 b23 r60	b51 b21 r59	b50 b16 r58	b47 b15 r57	b46 b14 r56	b44 b12 r54	b43 b11 r53	b41 b10 r51	b39 b9 r50	b38 b8	b37 b6
c57	b62 b34 b4	b58 b31 b2	b57 b27 b1	b54 b25 r63	b53 b24 r62	b51 b22 r61	b50 b20 r59	b49 b15 r58	b46 b14 r57	b45 b13 r56	b43 b11 r55	b42 b10 r53	b40 b9 r52	b38 b8 r50	b37 b7	b36 b5
c56	b61 b33 b3	b57 b30 b1	b56 b26 b0	b53 b24 r62	b52 b23 r61	b50 b21 r60	b49 b19 r58	b48 b14 r57	b45 b13 r56	b44 b12 r55	b42 b10 r54	b41 b9 r52	b39 b8 r51	b37 b7 r49	b36 b6	b35 b4
c55	b60 b32 b2	b56 b29 b0	b55 b25 r61	b52 b23 r60	b51 b22 r59	b49 b20 r57	b48 b18 r56	b47 b13 r55	b44 b12 r54	b43 b11 r53	b41 b9 r51	b40 b8 r50	b38 b7 r48	b36 b6	b35 b5	b34 b3
c54	b59 b31 b1	b55 b28 r60	b54 b24 r59	b51 b22 r58	b50 b21 r56	b48 b19 r55	b47 b17 r54	b46 b12 r53	b43 b11 r52	b42 b10 r50	b40 b8 r49	b39 b7	b37 b6	b35 b5	b34 b4	b33 b2
c53	b58 b30 b0	b54 b27 r59	b53 b23 r58	b50 b21 r57	b49 b20 r55	b47 b18 r54	b46 b16 r53	b45 b11 r52	b42 b10 r51	b41 b9 r49	b39 b7 r48	b38 b6	b36 b5	b34 b4	b33 b3	b32 b1
c52	b57 b29 r63	b53 b26 r58	b52 b22 r57	b49 b20 r56	b48 b19 r54	b46 b17 r53	b45 b15 r52	b44 b10 r51	b41 b9 r50	b40 b8 r48	b38 b6	b37 b5	b35 b4	b33 b3	b32 b2	b31 b0
c51	b63 b16 r50	b59 b15 r49	b56 b12 r48	b55 b11	b48 b9	b47 b4	b45 b3	b41 b2	b40 b1	b39 b0	b35 r63	b34 r60	b32 r59	b25 r57	b19 r52	b18 r51
c50	b62 b15 r49	b58 b14 r48	b55 b11 b10	b54 b10	b47 b8	b46 b3	b44 b2	b40 b1	b39 b0	b38 r63	b34 r62	b33 r59	b31 r58	b24 r56	b18 r51	b17 r50
c49	b61 b14	b57 b13	b54 b10	b53 b9	b46 b7	b45 b2	b43 b1	b39 b0	b38 r62	b37 r61	b33 r58	b32 r57	b30 r55	b23 r50	b17 r49	b16 r48
c48	b60 b13	b56 b12	b53 b9	b52 b8	b45 b6	b44 b1	b42 b0	b38 r63	b37 r61	b36 r60	b32 r57	b31 r56	b29 r54	b22 r49	b16 r48	b15

**Table A.5 – 80-bit LCRC checker equations**

Output	Exclusive OR these bits together															
c63	b79 b30 r63	b75 b28 r62	b71 b27 r61	b68 b24 r60	b67 b23 r57	b60 b21 r55	b59 b16 r53	b57 b15 r52	b53 b14 r50	b52 b13 r50	b51 b12 r50	b47 b9 r50	b46 b7 r50	b44 b5 r50	b37 b4 r50	b31 b2 r50
c62	b78 b29 r63	b74 b27 r62	b70 b26 r61	b67 b23 r60	b66 b22 r59	b59 b20 r56	b58 b15 r54	b56 b14 r52	b52 b13 r51	b51 b12 r49	b50 b11 r49	b46 b8 r49	b45 b6 r49	b43 b4 r49	b36 b3 r49	b30 b1 r49
c61	b77 b28 r62	b73 b26 r61	b69 b25 r60	b66 b22 r59	b65 b21 r58	b58 b19 r55	b57 b14 r53	b55 b13 r51	b51 b12 r50	b50 b11 r48	b49 b10 r48	b45 b7 r48	b44 b5 r48	b42 b3 r48	b35 b2 r48	b29 b0 r48
c60	b76 b27 r60	b72 b25 r59	b68 b24 r58	b65 b21 r57	b64 b20 r54	b57 b18 r52	b56 b13 r50	b54 b12 r49	b50 b11 r49	b49 b10 r49	b48 b9 r49	b44 b6 r49	b43 b4 r49	b41 b2 r49	b34 b1 r49	b28 r61
c59	b75 b26 r59	b71 b24 r58	b67 b23 r57	b64 b20 r56	b63 b19 r53	b56 b17 r51	b55 b12 r49	b53 b11 r48	b49 b10 r48	b48 b9 r48	b47 b8 r48	b43 b5 r48	b42 b3 r48	b40 b1 r48	b33 b0 r48	b27 r60
c58	b79 b51 r59	b75 b48 r58	b74 b44 r56	b71 b42 r53	b70 b41 r48	b68 b39 r48	b67 b37 r48	b66 b32 r48	b63 b31 r48	b62 b30 r48	b60 b28 r48	b59 b27 r48	b57 b26 r48	b55 b25 r48	b54 b24 r48	b53 b22 r48
c57	b78 b50 r57	b74 b47 r55	b73 b43 r52	b70 b41 r52	b69 b40 r52	b67 b38 r52	b66 b36 r52	b65 b31 r52	b62 b30 r52	b61 b29 r52	b59 b27 r52	b58 b26 r52	b56 b25 r52	b54 b24 r52	b53 b23 r52	b52 b21 r52
c56	b77 b49 r56	b73 b46 r54	b72 b42 r51	b69 b40 r51	b68 b39 r51	b66 b37 r51	b65 b35 r51	b64 b30 r51	b61 b29 r51	b60 b28 r51	b58 b26 r51	b57 b25 r51	b55 b24 r51	b53 b23 r51	b52 b22 r51	b51 b20 r51
c55	b76 b48 r56	b72 b45 r55	b71 b41 r53	b68 b39 r50	b67 b38 r50	b65 b36 r50	b64 b34 r50	b63 b29 r50	b60 b28 r50	b59 b27 r50	b57 b25 r50	b56 b24 r50	b54 b23 r50	b52 b22 r50	b51 b21 r50	b50 b19 r50
c54	b75 b47 r56	b71 b44 r55	b70 b40 r54	b67 b38 r52	b66 b37 r49	b64 b35 r49	b63 b33 r49	b62 b28 r49	b59 b27 r49	b58 b26 r49	b56 b24 r49	b55 b23 r49	b53 b22 r49	b51 b21 r49	b50 b20 r49	b49 b18 r49
c53	b74 b46 r55	b70 b43 r54	b69 b39 r53	b66 b37 r51	b65 b36 r48	b63 b34 r48	b62 b32 r48	b61 b27 r48	b58 b26 r48	b57 b25 r48	b55 b23 r48	b54 b22 r48	b52 b21 r48	b50 b20 r48	b49 b19 r48	b48 b17 r48
c52	b73 b45 r54	b69 b42 r53	b68 b38 r52	b65 b36 r50	b64 b35 r50	b62 b33 r50	b61 b31 r50	b60 b26 r50	b57 b25 r50	b56 b24 r50	b54 b22 r50	b53 b21 r50	b51 b20 r50	b49 b19 r50	b48 b18 r50	b47 b16 r50
c51	b79 b32 b2	b75 b31 b1	b72 b28 r61	b71 b27 r59	b64 b25 r57	b63 b20 r56	b61 b19 r54	b57 b18 r51	b56 b17 r50	b55 b16 r49	b51 b13 r49	b50 b11 r49	b48 b9 r49	b41 b8 r49	b35 b6 r49	b34 b3 r49
c50	b78 b31 b1	b74 b30 b0	b71 b27 r63	b70 b26 r60	b63 b24 r58	b62 b19 r56	b60 b18 r55	b56 b17 r53	b55 b16 r50	b54 b15 r49	b50 b12 r48	b49 b10 r48	b47 b8 r48	b40 b7 r48	b34 b5 r48	b33 b2 r48
c49	b77 b30 b0	b73 b29 r63	b70 b26 r62	b69 b25 r59	b62 b23 r57	b61 b18 r55	b59 b17 r54	b55 b16 r52	b54 b15 r49	b53 b14 r48	b49 b11 r48	b48 b9 r48	b46 b7 r48	b39 b6 r48	b33 b4 r48	b32 b1 r48
c48	b76 b29 r63	b72 b28 r62	b69 b25 r61	b68 b24 r58	b61 b22 r56	b60 b17 r54	b58 b16 r53	b54 b15 r51	b53 b14 r48	b52 b13 r48	b48 b10 r48	b47 b8 r48	b45 b6 r48	b38 b5 r48	b32 b3 r48	b31 b0 r48

**Table A.6 – 64-bit ECRC generator / checker equations**

Output	Exclusive OR these bits together															
c47	b63 b22 r42	b59 b20 r39	b55 b19 r38	b51 b18 r36	b50 b17 r35	b48 b13 r34	b43 b11 r33	b42 b10 b7	b40 b6 b4	b37 b6 b4	b35 b4 b3	b34 b3 b2	b32 b2 b1	b31 b1 r45	b29 r45	b26 r43
c46	b63 b35 b7	b62 b33 b5	b59 b32 b4	b58 b30 b0	b55 b29 r45	b54 b28 r44	b51 b26 r43	b49 b25 r41	b48 b22 r39	b47 b21 r37	b43 b20 r36	b41 b16 r32	b40 b13 b12	b39 b12 b11	b37 b11 b9	b36 b9
c45	b62 b34 b6	b61 b32 b4	b58 b31 b3	b57 b29 r47	b54 b28 r44	b53 b27 r43	b50 b25 r42	b48 b24 r40	b47 b21 r38	b46 b20 r36	b42 b19 r35	b40 b15 b12	b39 b12 b11	b38 b11 b10	b36 b10 b8	b35 b8
c44	b63 b43 b14	b61 b42 b13	b60 b41 b9	b59 b40 b6	b57 b39 b5	b56 b38 b4	b55 b33 b1	b53 b32 r46	b52 b30 r45	b51 b29 r41	b50 b28 r38	b49 b27 r37	b48 b24 r36	b47 b23 r33	b46 b22 r33	b45 b17
c43	b62 b42 b13	b60 b41 b12	b59 b40 b8	b58 b39 b5	b56 b38 b4	b55 b37 b3	b54 b32 b0	b52 b31 r45	b51 b29 r44	b50 b28 r40	b49 b27 r37	b48 b26 r36	b47 b23 r35	b46 b22 r32	b45 b21 r32	b44 b16
c42	b61 b41 b12	b59 b40 b11	b58 b39 b7	b57 b38 b4	b55 b37 b3	b54 b36 b2	b53 b31 r47	b51 b30 r44	b50 b28 r43	b49 b27 r39	b48 b26 r36	b47 b25 r35	b46 b22 r34	b45 b21 r34	b44 b20 r34	b43 b15
c41	b60 b40 b11	b58 b39 b10	b57 b38 b6	b56 b37 b3	b54 b36 b2	b53 b35 b1	b52 b30 r46	b50 b29 r43	b49 b27 r42	b48 b26 r38	b47 b25 r35	b46 b24 r34	b45 b21 r33	b44 b20 r33	b43 b19 r33	b42 b14
c40	b59 b39 b10	b57 b38 b9	b56 b37 b5	b55 b36 b2	b53 b35 b1	b52 b34 b0	b51 b29 r45	b49 b28 r42	b48 b26 r41	b47 b25 r37	b46 b24 r34	b45 b23 r33	b44 b20 r32	b43 b19 r32	b42 b18 r32	b41 b13
c39	b58 b38 b9	b56 b37 b8	b55 b36 b4	b54 b35 b1	b52 b34 b0	b51 b33 r44	b50 b28 r41	b48 b27 r40	b47 b25 r36	b46 b24 r33	b45 b23 r32	b44 b22 r32	b43 b19 r32	b42 b18 r32	b41 b17 r32	b40 b12
c38	b57 b37 b8	b55 b36 b7	b54 b35 b3	b53 b34 b0	b51 b33 r43	b50 b32 r40	b49 b27 r39	b47 b26 r35	b46 b24 r32	b45 b23 r32	b44 b22 r32	b43 b21 r32	b42 b18 r32	b41 b17 r32	b40 b16 r32	b39 b11
c37	b56 b36 b7	b54 b35 b6	b53 b34 b2	b52 b33 r47	b50 b32 r42	b49 b31 r39	b48 b26 r38	b46 b25 r34	b45 b23 r34	b44 b22 r34	b43 b21 r34	b42 b20 r34	b41 b17 r34	b40 b16 r34	b39 b15 r34	b38 b10
c36	b55 b35 b6	b53 b34 b5	b52 b33 b1	b51 b32 r47	b49 b31 r46	b48 b30 r41	b47 b25 r38	b45 b24 r37	b44 b22 r33	b43 b21 r33	b42 b20 r33	b41 b19 r33	b40 b16 r33	b39 b15 r33	b38 b14 r33	b37 b9
c35	b63 b24 b0	b59 b23 r47	b55 b22 r46	b54 b21 r43	b52 b17 r42	b47 b15 r40	b46 b14 r39	b44 b11 r38	b41 b10 r37	b39 b8 r35	b38 b7 r34	b36 b6 r33	b35 b5 r32	b33 b3 r32	b30 b2 r32	b26 b1
c34	b62 b23 r46	b58 b22 r45	b54 b21 r42	b53 b20 r41	b51 b16 r39	b46 b14 r38	b45 b13 r37	b43 b10 r36	b40 b9 r34	b38 b7 r33	b37 b6 r32	b35 b5 r32	b34 b4 r32	b32 b2 r32	b29 b1 r32	b25 b0
c33	b61 b22 r45	b57 b21 r44	b53 b20 r41	b52 b19 r40	b50 b15 r38	b45 b13 r37	b44 b12 r36	b42 b9 r35	b39 b8 r33	b37 b6 r32	b36 b5 r32	b34 b4 r32	b33 b3 r32	b31 b1 r32	b28 b0 r32	b24 r47
c32	b60 b21 r43	b56 b20 r40	b52 b19 r39	b51 b18 r37	b49 b14 r36	b44 b12 r35	b43 b11 r34	b41 b8 r32	b38 b7 r32	b36 b5 r32	b35 b4 r32	b33 b3 r32	b32 b2 r32	b30 b0 r32	b27 r46	b23 r44

